Inference under the coalescent with recombination

by

ALI MAHMOUDI

ORCID: 0000-0003-3813-3477

Doctor of Philosophy

March 2021

Faculty of Science

School of Mathematics and Statistics

Melbourne Integrative Genomics

The thesis is being submitted in total fulfilment of the degree.
The degree is not being completed under a jointly awarded degree.

THE UNIVERSITY OF MELBOURNE

Abstract

Faculty of Science
School of Mathematics and Statistics

Doctor of Philosophy

Inference under the coalescent with recombination

by ALI MAHMOUDI

Inferring the genealogical history, also known as the Ancestral Recombination Graph (ARG), of a set of DNA sequences has been a central challenge in population genetics for decades. Reconstructing the actual ARG simplifies many inference problems in population genetics. Many different methods have been proposed for inferring the ARG, most of which are limited in size and accuracy. The state-of-the-art probabilistic model, *ARGweaver*, provides substantial improvements over other methods but uses a discretized version of the Sequentially Markov Coalescent (SMC), which is an approximation of the Coalescent with Recombination (CwR) and ignores a significant amount of information in the ARG.

In this thesis, I develop a novel Markov Chain Monte Carlo (MCMC) algorithm, implemented in the software *ARGinfer*, to perform probabilistic inference under the CwR. This method takes advantage of the superior properties of the Tree Sequence (TS), which is an efficient data structure to store the genealogical trees in an ARG so that the identical subtrees of the neighboring trees are recorded only once. I first devise a data structure to represent the ARG and the mutation information by augmenting the TS. Then, I develop a heuristic algorithm to construct an ARG consistent with the data used as an initial value for the MCMC algorithm. Computing both the prior (CwR model) and the likelihood under an approximation to the infinite sites model are relatively straightforward and fast. The challenging part is to explore the ARG space, for which I introduce a proposal distribution in the form of six transition types to rearrange both the topology and the event times.

I demonstrate the utility of *ARGinfer* by applying it to simulated data sets. *ARGinfer* can accurately estimate many ARG-derived parameters such as the total branch length, number of recombination events, time to the most recent common ancestor, recombination rate, and allele ages. I also compare *ARGinfer* against *ARGweaver*. Since *ARGinfer* assumes a more complex evolutionary model than *ARGweaver*, it can infer a larger class of parameters. *ARGinfer* outperforms *ARGweaver* in estimating the recombination rate and is at least as accurate for other parameters that *ARGweaver* can infer. *ARGinfer* also accurately estimates parameters that *ARGweaver* cannot, such as the number of recombinations on trapped non-ancestral materials.

Declaration of Authorship

I, ALI MAHMOUDI, declare that this thesis titled, "Inference under the coalescent with recombination" and the work presented in it are my own. I confirm that:

- The thesis comprises only my original work towards the Doctor of Philosophy degree except where indicated in the Preface;
- due acknowledgement has been made in the text to all other material used; and
- this thesis is fewer than 100,000 words in length, exclusive of tables, maps, bibliographies, and appendices.

Signed:

Date:

Preface

The idea of the data structure introduced in chapter 5 is a result of multiple discussions and correspondences with Dr. Jere Koskela from the University of Warwick and Dr. Jerome Kelleher from the University of Oxford.

All the work in this thesis is original and entirely carried out by Ali Mahmoudi. None of the work in this thesis has been submitted for other qualifications. None of the work in this thesis was carried out before enrollment in the degree. No third party editorial assistance was provided in the preparation of this thesis.

I am supported by the Melbourne Research Scholarship (MRS) and Xing Lei Scholarship.

Acknowledgements

Undertaking this Ph.D. has been a unique and genuinely life-changing experience for me, and it would not have been possible to do without the support, guidance, and encouragement of many people.

I express my sincere appreciation to my principal supervisor Prof. David Balding, who dearly supported me all the time, encouraging and guiding my work in the right direction. I have been fortunate to have had the opportunity to work with David in the past three and a half years. His advice on both research as well as personal matters has been priceless. Without his guidance and constant feedback, this Ph.D. would not have been achievable.

I am deeply grateful to my co-supervisor, Dr. Yao-ban Chan, who has been an excellent and dedicated supervisor. Yao-ban has been a great inspiration, and I have greatly benefited from his ideas, time, and feedback. He is a supportive, smart, and passionate scientist, and I am very proud to be his Ph.D. student.

I would like to thank my Ph.D. committee chair Dr. Nathan Ross for his expertise, guidance, and support during my candidature.

My thanks also go out to the support I received from my collaborators Dr. Jere Koskela, the University of Warwick, and Dr. Jerome Kelleher, the University of Oxford. I am incredibly grateful for the fruitful discussion and valuable advice, both professional and personal.

I gratefully acknowledge the funding received towards my Ph.D. from the Melbourne Research Scholarship (MRS) and Xing Lei Scholarship, the University of Melbourne. I also acknowledge the Faculty of Science Abroad Traveling Scholarship and the Belz fund from the school of Mathematics and Statistics for supporting me through my visit to the Big Data Institute at Oxford and the "probabilistic modeling in genomics" conference in France.

I am also very grateful to all those at the Melbourne Integrative Genomics (MIG), especially Dr. Andrew Siebel, Mr. Bobbie Shaban, and others who were always so helpful and provided me with their support throughout my candidature.

I would like to express my appreciation to my fellow Ph.D. colleagues and friends, Jihye D. Shin, Georgia Tsambos, Anubhav Kaphle, Katalina Bobowik, Abolfazl JalalAbadi, Yiwen Wang, Qian Feng, Chengyu Li, Yupei You, and Qiuyi Li for their moral support and encouragement during my Ph.D. journey.

I would like to thank my parents and siblings. Their support, care, and love during this time have been overwhelming. Without you in my life, this would not have been possible.

Finally, many thanks to everyone whom I could not mention personally, but have helped in various forms or ways to make this Ph.D. a memorable experience and less stressful.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The rise of advanced sequencing technologies has made a vast wealth of genetic data available. Such data provide a rich source of information to address a variety of questions concerning human history, human origin, and evolutionary forces, such as: What are the mutation and recombination rates? What is the effective population size through time? How did the genome change over time? What is the time to the most recent common ancestor? Knowledge about such evolutionary forces and a good understanding of patterns of variation in our DNA is central to study the genetic bases of human diseases, demographic history, and biological processes. Hence, it is important to develop methods to obtain useful information from genetic data.

One of the powerful models to describe the genealogical relationships of a sample of individuals was introduced by Kingman (1982a,b), known as Kingman's coalescent or (in brief) the coalescent. The coalescent revolutionized population genetics, offering crucial insights into how different evolutionary forces affect the genetic structure of contemporary data. Hudson (1983) modified the coalescent to include recombination, called the coalescent with recombination (CwR), and designed an algorithm to simulate the CwR. A mathematically equivalent model to the CwR is the Ancestral Recombination Graph (ARG) (Griffiths and Marjoram, 1997). The term "ARG" is treated in two ways in the literature; a stochastic process, i.e., the CwR, and a realization of the process, which is a fixed graph structure. We use the term "ARG" for a single realization of the CwR, i.e., a fixed graph.

The ARG describes the evolutionary relationships among a collection of related genome sequences. At each genomic site, there is a genealogical tree embedded in the ARG. More precisely, an ARG details genetic transmission events that occurred across generations in an evolving population. This graph includes features that are useful starting points for many population genetics analyses and simplifies many statistical inference problems, such as recombination and mutation rates and effective population size estimates. ARGs also facilitate local ancestry demography inference problems and detect sequences under selection (Hubisz, 2019; Arenas, 2013).

Inferring the ARG has been a central challenge in population genetics for decades. It is hard because, firstly, the space of possible ARGs involves a high-dimensional product of continuous and finite spaces and is astronomically large such that a comprehensive iteration even over the finite spaces is infeasible. Secondly, information about recombination in the observed data is

often insufficient to prefer one ARG to others, i.e., many topologically different ARGs have similar likelihoods. Due to these complexities, ARGs have not been widely used, and inference is often based on summary statistics that, by their nature, discard some information.

However, some statistical techniques have been used to infer the ARG of a sample of observed chromosomes. Early efforts to tackle this problem mainly used importance sampling, or Markov Chain Monte Carlo (MCMC) methods, e.g. see (Griffiths and Tavaré, 1994; Stephens and Donnelly, 2000; Fearnhead and Donnelly, 2001; Jenkins, 2008; Kuhner et al., 1995). Although they made significant contributions to the field, these models scale poorly with sample size and sequence length and are restricted to a small number of sequences (see Chapter 2).

McVean and Cardin (2005) introduced an approximation to the CwR that reduces the state space of the ARG. This approximation is the Sequentially Markov Coalescent (SMC) and assumes that the genealogical trees at each site are Markov along the genome. More recently, Rasmussen et al. (2014) developed an MCMC algorithm to infer the ARG of a sample of DNA sequences. This model assumes an approximation of the SMC that discretizes the event times and genomic sites. The algorithm is implemented in *ARGweaver*, which can handle a few tens of sequences genome-wide.

In an ARG, the genealogical trees at neighboring genome sites share many subtrees. As we move along the sequence, the topologies of the consecutive genealogical trees change according to the impact of the recombination events. A recombination event often affects a handful of tree branches and leave the remaining subtrees unchanged, resulting in a high similarity between adjacent trees (see section 3.1). Before 2016, this fact has been neglected in ARG representation. A key breakthrough occurred when Kelleher et al. (2016) developed a data structure known as the Tree Sequence (TS) to store and represent genealogical trees so that identical subtrees are not repeated. The software developed by them, *msprime*, can efficiently simulate the genealogical history of millions of sequences under the CwR. This simulator is significantly faster (in terms of time and memory) than existing methods, including those based on approximations of the CwR.

The TS data structure captured the attention of population genetics researchers for its efficiency in analyzing genealogies. By monitoring these successes in the simulation and processing of genealogies, questions about the inference problem under the CwR arise: Does TS facilitate the inference problem under the CwR? This thesis attempts to answer this question.

1.1 Aims and structure of this thesis

The overall aim of this thesis is to develop a novel MCMC algorithm that:

- benefits from the efficient properties of TS,
- works under a more accurate population model than the SMC.

To date, there has been no research on inferring the CwR by exploiting the efficient properties of TS. In this thesis, I first devise a new efficient data structure to represent ARGs. Then, I develop algorithms to construct ARGs, evaluate likelihood, and explore the ARG space, leading to an MCMC algorithm that assumes the CwR and samples from the posterior distribution of ARGs. Hence, various features such as the time to the most recent common ancestor, the allele age, the topology of the tree(s), the recombination and mutation rates, and the total branch length can be inferred.

The structure of this thesis is as follows. Chapter 2 reviews the literature. In this chapter, I discuss the theoretical basis and some key concepts needed to better understand the coalescent and inferring the ARG. The main focus is on the existing inference methods, from importance sampling to MCMC algorithms. More recent heuristic methods that tackle a significantly simpler problem, i.e., constructing a single ARG consistent with the data rather than inferring the posterior distribution, are also discussed.

In chapter 3, I discuss the motivations that inspired me to perform this research. I explain the efficiencies of the TS and the significance of trapped non-ancestral materials in ARGs.

In chapter 4, I first discuss that the TS does not include all information required in ARG inference. Then, I introduce a new data structure called the Full TS (FTS), which takes advantage of TS features. Then, I develop an MCMC algorithm to perform inference under the CwR. The MCMC algorithm has three steps. The first step is to construct an initial ARG for the given data, for which we devise a heuristic algorithm to build and represent an ARG in FTS format. The second step of the algorithm is to develop algorithms to calculate the likelihood, posterior, and transition probabilities. The last MCMC step is to define a proposal distribution to allow the algorithm to explore the ARG space. I conclude that the introduced MCMC algorithm is slow and does not scale well. The limitations and complexities of the ATS are therefore discussed.

Chapter 5 is dedicated to a new data structure that I developed to represent an ARG. This data structure is an augmentation of TS and is named Augmented TS (ATS). An MCMC algorithm under the ATS is developed, which is implemented in a Python software called *ARGinfer*.

In chapter 6, I evaluate the performance of *ARGinfer* using simulated data and show that *ARGinfer*'s outputs are an accurate approximation for the posterior distribution. This algorithm can infer ARG properties with high accuracy. I also compare *ARGinfer* against *ARGweaver*. The results show that for the features that *ARGweaver* can infer, *ARGinfer* performs at least as accurate as *ARGweaver*. *ARGinfer* is based on the CwR and therefore deals with a larger class of recombination events, and can accurately infer features that *ARGweaver* cannot, such as the number of non-ancestral recombinations.

Finally, in chapter 7 I summarize and conclude the thesis and present future directions that can be investigated.

Chapter 2

Review of coalescent modeling and inference

2.1 Introduction

This chapter studies the coalescent process that describes the evolution of DNA sequences backward in time. In section 2.2, we introduce the infinite sites model. Section 2.3 presents the Wright-Fisher model and the standard coalescent. The standard coalescent does not model recombination. In section 2.4, we discuss the CwR. The first formulation for the CwR is backward in time, which is explained in section 2.4.1. Section 2.4.2 introduces another approach to model the CwR sequentially along the genome. Section 2.5 is an overview of inference methods in three paradigms: importance sampling, heuristic, and MCMC techniques.

Note that throughout this thesis, I use "sequence" interchangeably with "haplotype", "chromosome" , or "gene".

2.2 The infinite sites model of mutation

Kimura (1969) introduced the infinite sites model (ISM) to approximate chromosomes by a continuous interval. Each time a mutation occurs, it affects a site that was previously unaffected. Therefore, a site always carries at most two states labeled as 0 and 1 with no biological meaning. It is enough to keep track of only the segregating sites as a series of zeros and ones since the remaining sites do not carry any mutation information. Therefore, a sample of n DNA sequences each with m Single Nucleotide Polymorphisms (SNP) can be represented as a binary matrix D with $D^i_j \in \{0, 1\}$ for $1 \leq i \leq n$ and $1 \leq j \leq m$.

In practice, chromosomes are finite in sites, and the ISM is never true. In this thesis, a discrete approximation of the ISM is used: We assume a finite number of sites where mutations can occur, and we only allow at most one mutation at each site.

2.3 From the Wright-Fisher model to the coalescent

For sequences within a finite population of a fixed size, Wright (1931) and Fisher (1930) introduced a model that prospectively describes the evolution

process. The Wright-Fisher model makes some simplifying assumptions, including

- discrete and non-overlapping generations,
- constant population size for all generations,
- no selection, and
- no recombination.

This model can be applied to haploid and diploid organisms. We assume the population is diploid, in which case for a population of N individuals, there will be $2N$ copies of each gene. The Wright-Fisher model assumes that the genes at generation $j+1$ are obtained by random sampling with replacement from generation j. With probability $1/2N$, a gene at generation j is copied from to create its offspring in generation $j+1$. This process continues from one generation to the next until the current generation.

The number of offspring of gene $i, i = 1, 2, ..., 2N$ at generation $j+1$ has a binomial distribution with parameters $2N$ and $1/2N$. For large N, the distribution of the number of descendants of gene i can be approximated by a Poisson distribution with mean 1. A gene does not leave any offspring with a $e^{-1} \approx 0.37$ probability. Hence, the Wright-Fisher model wastes computation on genes that do not have any descendants in the current generation. For instance, a population of size $2N = 10,000$ results from 99 ancestral genes about ten generation ago. Approximately 9901 genes in the population ten generations ago did not survive until the current generation.

An alternative is to model the evolution process backward in time. Kingman (1982a,b) introduced a model to describe the history of a sample of size n. The coalescent assumes that N is much larger than n ($n \ll N$). Going backward in time, when two lineages find a common ancestor in the past and merge, a *coalescence event* occurs, which decreases the number of lineages by one. The coalescence events between the lineages continue until only one sequence remains, which is a common ancestor for all the sequences in the sample and is called the *Most Recent Common Ancestor* (MRCA). The resulting structure is a tree that describes the ancestry of a sample and is referred to as the *genealogy*. A genealogy has two components, the topology (branching order) and the ages of the internal nodes (event times). The branching order and event times are independent, and at any time, any two lineages are equally likely to find a common ancestor so that all branching orders are equally likely.

If there are k lineages at the current generation, the probability of no coalescence event in the previous generation is

$$\prod_{i=1}^{k-1}(1-\frac{i}{2N}) = 1 - \binom{k}{2}\frac{1}{2N} + \mathcal{O}(\frac{1}{N^2}), \tag{2.1}$$

where the last term is all the terms divided by N to the power of 2 or higher. Since $N \gg n$, $\mathcal{O}(\frac{1}{N^2})$ is negligible and can be ignored. Therefore, the probability of having more than one coalescence event at the same generation is

assumed to be zero, and with probability

$$\binom{k}{2}\frac{1}{2N},$$

a coalescent event occurs in a given generation. As a consequence, the waiting time (T_k) for a coalescence event between any two genes out of k genes has a geometric distribution with parameter $\binom{k}{2}/2N$.

Kingman (1982a) proved that in the limit as $N \to \infty$, the genealogies under the Wright-Fisher model converge to the standard coalescent with continuous time. If the population is diploid, coalescent time is measured in units of $2N$ generations, i.e., $t = j/2N$, where j is the time measured in generations. With this formulation, the waiting time to the next coalescence event follows an exponential distribution with rate $\binom{k}{2}$. Figure 2.1 illustrates a realization of the coalescent with time measured in both generations and coalescent units.

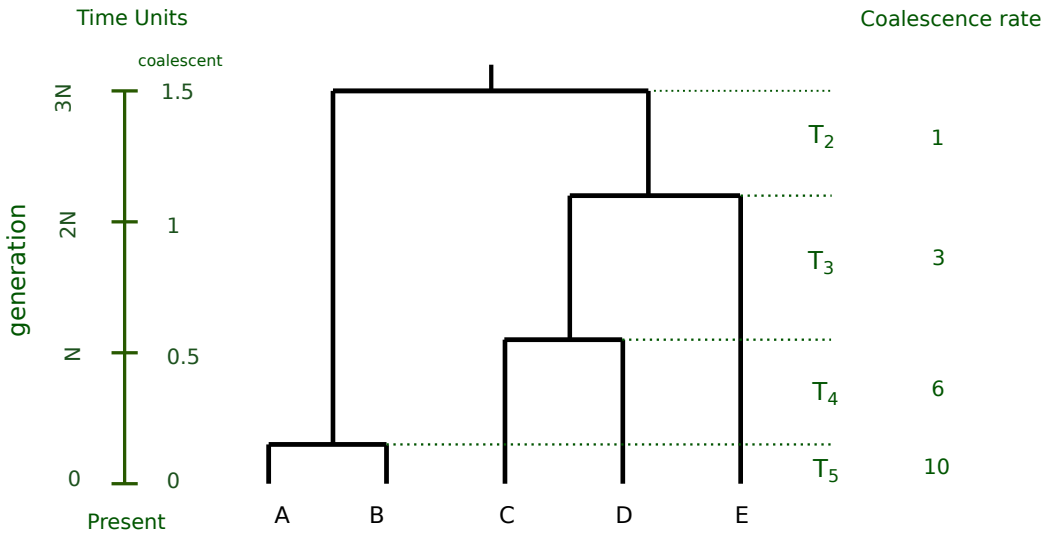


Figure 2.1: A realization of the coalescent for a sample of 5 sequences. On the left, the time in units of generations and coalescent is represented. On the right, the numbers represent the coalescence rates for the corresponding $T_k, k = 2, 3, 4, 5$.

An important property of the coalescent process is that the genealogy and the mutation process can be treated separately under neutrality. With this property, mutations can be superimposed on the separately-generated trees. The mutation process can be described by a Poisson process along the genealogy branches at rate $\theta/2$, where $\theta = 4N\mu$ is called the population mutation rate or the scaled mutation rate per site and μ is the mutation rate per site per generation.

In the next section, we show that the coalescent can be extended to include recombination, increasing the scope of the model to longer genome intervals.

2.4 The coalescent with recombination

The CwR is a stochastic process that adds recombination to the standard coalescent. The genealogical history can be different on either side of a recombination event, and so the relationships among a sample of sequences are described by a graph. There are two mathematically equivalent approaches to construct the CwR. The first approach is the Hudson model, which builds the process backward in time, and the second is the Wiuf and Hein model that constructs the process by moving along the genome. In other words, the former goes back in time and, at each time, builds the whole sequences. Whereas the latter goes one site at a time and makes the whole genealogy at each site.

2.4.1 Hudson model

Hudson (1983) proposed a method to generate the CwR. There are two possible events backward in time, either recombination or common ancestor (CA). A recombination event causes a sequence to split into two sequences from a randomly chosen point so that one of the newly created sequences carries the ancestral material to the left of the breakpoint, and the other carries the ancestral material to the right. The two sequences resulting from a recombination both include sites that are not ancestral to any observed sequences. One or both sequences may be entirely non-ancestral to the sample, called *No Ancestral Material* (NAM) sequences. Figure 2.2 illustrates four types of recombination events that may occur in an ARG, backward in time. A CA event merges two sequences; if the sequences share ancestral material, we call the event as *coalescence* event.

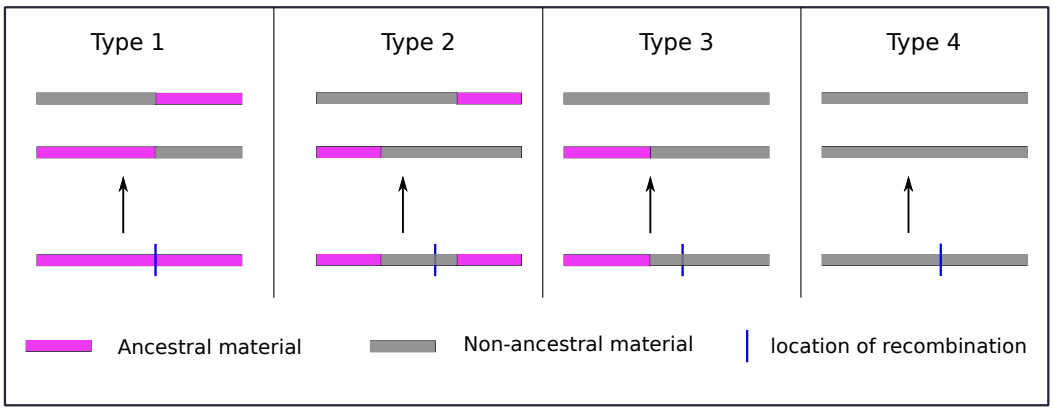


Figure 2.2: Recombination going backward in time from a sample of observed sequences. Purple indicates material that is ancestral to at least one observed sequence, and grey is non-ancestral material. The blue vertical line shows the recombination breakpoint. In Type 3, the right parent is a NAM lineage. Type 4 creates two NAM lineages because the child sequence is NAM.

The time to a recombination event is exponentially distributed with rate $k\rho/2$, where k is the number of lineages, $\rho = 4Nr$ is the scaled recombination rate per site, and r is the recombination rate per site per generation. The time

to a CA event is also exponentially distributed with rate $k(k-1)/2$. Therefore the time to the next event (backward in time) has an exponential distribution with rate

$$\lambda = \frac{k(k-1)}{2} + \frac{\rho k}{2}. \tag{2.2}$$

A basic algorithm to simulate a genealogy of n DNA sequences is as follows:

1. Start with $k = n$ sequences.
2. Simulate a time point from the exponential distribution with rate λ.
3. The event is either a CA or recombination with probability $k(k-1)/2$ and $k\rho/2$, respectively.
4. If it is a CA event, randomly choose two lineages and merge them. Decrease k by one. If $k = 1$ terminate the algorithm, otherwise go to step 2.
5. If it is a recombination event, choose a lineage and a breakpoint on the lineage at random. Create two lineages so that one carries the ancestral material to the left of the breakpoint, and the other carries the ancestral material to the right of the breakpoint. Increase k by one and go to step 2.

The remaining sequence is called the Grand MRCA (GMRCA). The resulting structure of this algorithm is an ARG (see Hein et al. 2004; Nordborg 2019).

For each genomic site, there is a coalescent tree embedded in the ARG called *the marginal tree*. To extract the marginal tree at a site from the ARG, one can track the genealogy at that site backward in time, starting from the sample sequences. Whenever a recombination event is encountered, follow the path of the parent that includes the followed site. Continue tracking the genealogy of the followed site until the MRCA is reached. The resulting binary tree is the marginal tree at the followed site.

We require some definitions. As is seen from Figure 2.2, both newly-created parents from recombination Type 1 and 2 carry ancestral material. These two recombination types contribute to the genetic structure of the current data (Wang et al., 2014). We refer to the Type 1 recombination as an *ancestral recombination* because the breakpoint occurs within ancestral material. A block of non-ancestral material trapped between two ancestral material segments is called *Trapped Non-ancestral Material (TNAM)*. If a recombination breakpoint occurs within a segment of TNAM, the recombination is referred to as a *non-ancestral recombination* (Type 2 in Figure 2.2).

The algorithm carries two unnecessary components. First, it simulates and tracks NAM lineages, which provide no information relevant to the observed data. These lineages can be avoided by tracking ancestral material and suppressing recombinations of type 3 and 4 in Fig 2.2. Second, the algorithm tracks all the sites of the sequences until the GMRCA, including sites that may have reached their MRCA long before the GMRCA. The algorithm can be adjusted by not tracking the materials for a site that has already reached MRCA.

Hudson modified the algorithm to consider the two factors mentioned above. The modified algorithm is called Hudson's algorithm, which is the most efficient algorithm to simulate the backward time CwR.

2.4.1.1 msprime

msprime, introduced by Kelleher et al. (2016), is a new implementation of Hudson's algorithm that enables simulation of large data sets. In *msprime*, when two ancestors merge, if they have overlapping ancestral material, i.e., a coalescence event, the event is recorded. The recorded information is called a *Coalescence Record* (CR). The CR includes the genomic interval of the overlapping ancestral material, the parent node, the two child nodes, and the time of the event.

The CR encoding is an efficient way to store the genealogies of DNA sequences and saves time and memory. The efficiency comes from storing the identical subtrees of the marginal trees only once. The importance of avoiding repetition is highlighted by the fact that any two consecutive trees of a genealogy share many similarities. Therefore, storing a marginal tree at every site, which was previously the standard practice, is inefficient.

To illustrate, we discuss a simple example borrowed from Kelleher et al. (2016). Figure 2.3 is an example of simulating the ancestry of four sequences with ten sites using the CR. Panel (a) represents the four sequences, the event time (t), and the corresponding tree at the bottom. On the top, k' is the total number of available recombination links, which is a gap between sites where a recombination can occur; here, there are 4 sequences and 10 sites, so $k' = 4 \times (10 - 1) = 36$.

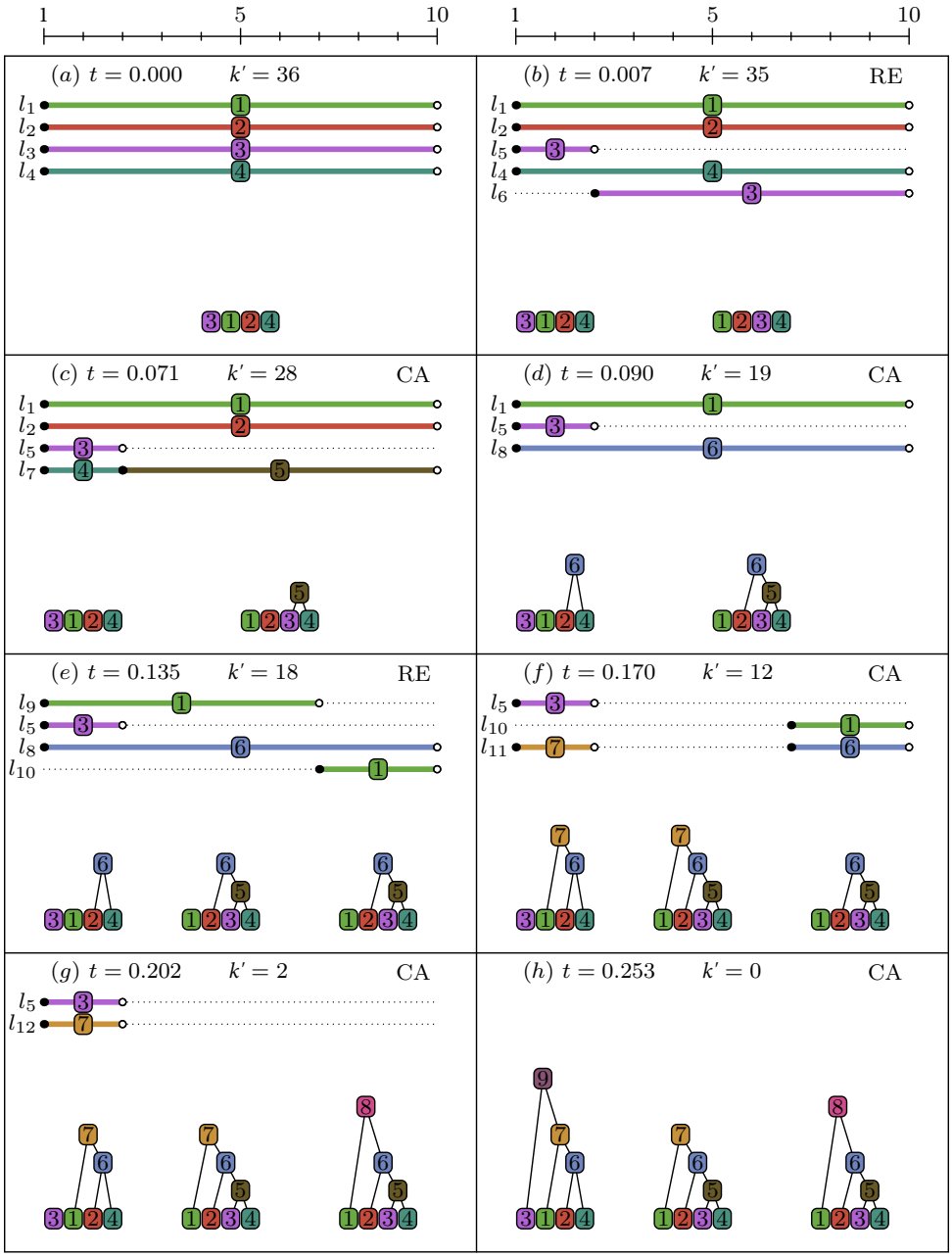


Figure 2.3: Hudson's algorithm using *msprime*. Each panel represents a state of the algorithm, see text for more information. This Figure is taken from Kelleher et al. (2016).

The first event that occurs moving backward in time is a recombination event at time $t = 0.007$ (panel (b)) splitting the sequence l_3 into l_5 and l_6 with ancestral material in interval $[1, 2]$ and $[3, 10]$ respectively. The next event is coalescence between l_4 and l_6, with overlapping material at interval $[3, 10]$ with 5 as the parent node. This event is recorded as $(3, 10, (3, 4), 5)$. The next event is a coalescence between l_7 and l_2 for which two intervals are recorded. The first record is $(1, 2, (2, 4), 6)$ due to overlapping material at interval $[1, 2]$, and the second is $(3, 10, (2, 5), 6)$. This process continues until all the genealogies

have been generated across the entire sequence. The output is presented in Table 2.1.

Table 2.1: The CR for the simulation in Figure 2.3. Each row is a coalescence record.

ID	left	right	parent	child	t
1	3	10	5	(3, 4)	0.071
2	1	2	6	(2, 4)	0.090
3	3	10	6	(2, 5)	0.090
4	1	7	7	(1, 6)	0.170
5	8	10	8	(1, 6)	0.202
6	1	2	9	(3,7)	0.253

msprime is significantly faster (in terms of time and memory) than other existing simulators including *msms* (Ewing and Hermisson, 2010), *msHOT* (Hellenthal and Stephens, 2007), *MaCS* (Chen et al., 2009), *SCRM* (Staab et al., 2015), and *ms* (Hudson, 2002), for large sample sizes, both in simulating coalescent trees and generating haplotypes. See Kelleher et al. (2016) for a comparison study.

Kelleher et al. (2018) modified and generalized the CR to be stored in a collection of tables: *nodes*, *edges*, *sites*, and *mutations*, which all together we call *Tree Sequence* (TS). The first two tables record the topology of the marginal trees, and the other two tables record the sequence information for each sample. The *nodes* table records an ancestor's birth time. The relationship between the nodes is presented in the *edges* table. For a specific genomic interval $[l, r)$ the parent node and a child node of the coalescence event is recorded. The *sites* table records the genome site and the ancestral state. Lastly, the *mutations* table records the SNP at which the mutation occurs, the (first) node on which the mutation has taken place, and the derived alleles (Kelleher et al., 2018).

Kelleher et al. (2019) showed that for recording millions of human-like simulated sequences, the TS requires remarkably less storage space that the VCF format (Danecek et al., 2011), which is the most widely used format. For instance, for 10 billion simulated sequences with 100 million sites, the VCF requires 25 PB, whereas the TS only needs 1 TB, which is $25,000$ times smaller.

Aside from efficient encoding of the genealogical histories, the TS provides a powerful platform to process and analyze the genealogical histories. Kelleher and colleagues designed a Python API that provides a general-purpose toolkit, called *tskit*[1], for importing, manipulating, processing, and outputting TS. *tskit* is under development and already contains many methods to process and analyze the tree sequences. Some methods compute statistics directly from tree sequences, without a need to call the genotypes. Refer to Ralph et al. (2020) for more detail on the methods to compute statistics using TS.

[1]https://tskit.readthedocs.io/en/latest/

2.4.2 Wiuf and Hein model

Another approach to describe the history of a sample is by moving along the sequence and adjusting the genealogy as we encounter recombination breakpoints. Wiuf and Hein (1999) introduced an algorithm to do so, which is called the *spatial algorithm* (the genome is considered as a spatial dimension) and is described in the following:

1. Simulate the genealogy of the leftmost site of the sequence under the standard coalescent. The tree is called the *local tree* at site zero. Denote the total branch length of the tree as A.

2. Simulate a genome location x from an exponential distribution with rate A. If x is within the sequence length, uniformly choose a time point on the local tree (or local graph, i.e., the built graph for the sites from the leftmost to the current site) and place a recombination at that point. Sites to the left of x share the same local graph, and the newly created lineage that carries the materials to the right of x coalesces with a lineage at random. Denote L as the sequence length. If $x > L$, terminate the algorithm.

3. Move the starting point to x and take A as the total branch length of the local graph at x. Go to step 2.

The spatial algorithm requires the knowledge of all the previous local trees (moving rightwards) to coalesce a newly created lineage by recombination to the local graph, and so is not Markovian.

The algorithm models the NAM lineages. It is not known in advance whether a recombination event contributes to the genealogy or not. It is only known after the algorithm is terminated. Hence, the spatial algorithm is slower than the Hudson algorithm (Hein et al., 2004). Approximating the process is one of the strategies to improve model efficiency.

2.4.2.1 The sequentially Markov coalescent

McVean and Cardin (2005) introduced an approach, called the sequentially Markov coalescent (SMC), to approximate the CwR, ignoring the non-Markovian property of the Wiuf and Hein model. The non-Markovian property of the spatial algorithm comes from CA events where the child sequences do not have any overlapping ancestral material. The main idea of the SMC is to construct a process in which such events are banned. The resulting process is, therefore, Markovian.

McVean and Cardin (2005) also devised the following algorithm to simulate the marginal genealogies on the unit interval for n DNA sequences under the SMC.

1. At genome location $x_0 = 0$ simulate a standard coalescent tree and take its total branch length as A.

2. Generate x' from the exponential distribution with rate $\rho A/2$. If $x = x' + x_0 < 1$, uniformly choose a time t_R on the marginal tree at x_0 as the time

at which recombination occurs and erase the older portion of the branch on which the event occurred, leading to a floating lineage.

3. The floating lineage coalesces with the remaining genealogy with rate $\sum_{i \neq j} I_{ij}$, where $I_{ij} = 1$ for those pair of lineages that have overlapping ancestral material; otherwise, $I_{ij} = 0$. The coalescence can occur any time before t_R and can be older than the current MRCA.

4. Take $x_0 = x$ and repeat steps 2 and 3 until $x > 1$.

Banning any CA event that would create TNAM reduces the state space of the ARG. An ARG under the SMC does not include any non-ancestral recombination (Type 2 in Figure 2.2). However, the distribution of marginal trees under the SMC and the CwR are the same (McVean and Cardin, 2005).

2.4.2.2 The SMC'

Marjoram and Wall (2006) introduced another Markovian model to approximate the CwR. The model is similar to the SMC and is called the *SMC'*. To explain the SMC', let us first define *invisible* recombination. Assume that lineage a undergoes recombination and creates two lineages b and c (backward in time). If b and c coalesce back together, the recombination is called *invisible*. The only difference between the SMC and SMC' is that the SMC' models invisible recombinations (see Figure 2.4). Thus, the SMC' is a more accurate approximation to the CwR than the SMC. Wilton et al. (2015) compared the joint distribution of coalescence times at two sites for the CwR, SMC, and SMC'. They showed that the difference between the CwR and the SMC' is significantly less than that of the CwR and SMC. Hence, since the SMC' is not more complicated than the SMC, it is better to use the SMC' approximation in simulation and inference methods.

2.5 Inferring the CwR

We wish to draw inferences about the population processes that generated a sample of n DNA sequences (D). Traditionally, inference of the rates θ and ρ under the coalescent has been limited to point estimation. For instance, Wakeley's estimator (Wakeley, 1997) of the scaled recombination rate, and Watterson's (Watterson, 1975) and Tajima's estimators (Tajima, 1983) of the scaled mutation rate. The major drawback of these estimators is that, unlike likelihood-based methods, they do not use all the information in the sample.

It is hard to calculate the full likelihood ($L(\Theta)$) of $\Theta = (\theta, \rho)$ because the history of the sequences is missing. It is useful to integrate over all possible ARGs (G) to calculate the likelihood accurately. The likelihood is defined as

$$L(\Theta) = P(D; \Theta) = \int_G P(D|G; \Theta) P(G; \Theta)\, dG. \tag{2.3}$$

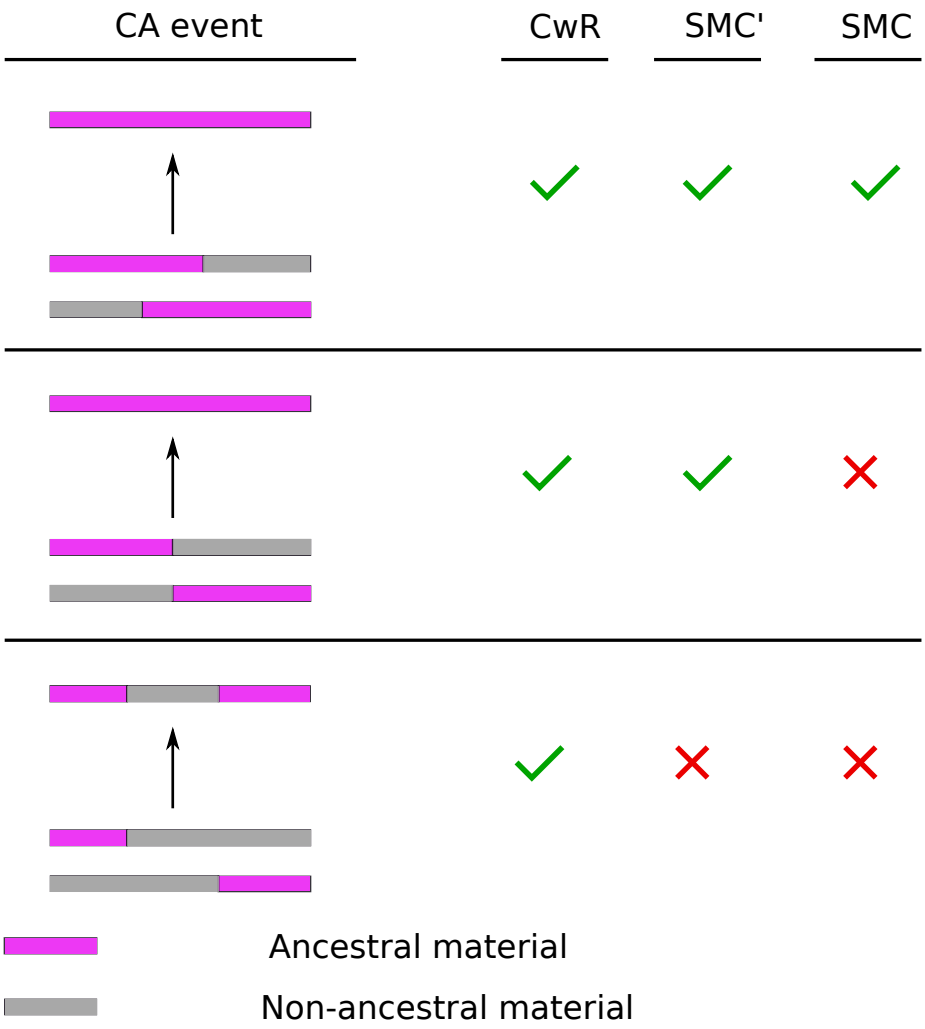


Figure 2.4: Three types of CA events. The top event is a CA where the child lineages contain overlapping ancestral material (a coalescence event). This event is allowed in all three models. The second CA event is between two lineages with adjacent ancestral material. This event is allowed in the CwR and SMC', but not the SMC. The last event is a CA between two lineages containing no overlapping ancestral material, which is only allowed in the CwR.

However, the integral in equation (2.3) is challenging to calculate analytically. Consequently, approximation methods are proposed for $L(\Theta)$ such as

$$L(\Theta) \approx \frac{1}{M} \sum_{i=1}^{M} P(D|G_i; \Theta), \tag{2.4}$$

where $G_1, G_2, ..., G_M$ are realizations of the CwR (Hein et al., 2004). This approximation is not efficient for samples of more than a few sequences, because most of the simulated ARGs are not compatible with D, i.e., $P(D|G_i; \theta) = 0$.

Realizations of the CwR are the ARG for a sample of n sequences ignoring the allelic types, which can be regarded as a prior distribution. With this prior distribution, we refer to the posterior distribution after observing D as the CwR+.

2.5.1 Importance sampling

One way to improve the estimator in equation (2.4) is to simulate the ARGs from a proposal distribution $Q(G)$ that is closer to the CwR+. $Q(G)$ can be constructed so that it takes account of the observed data. Ideally, $Q(G)$ gives non-zero probability to the compatible ARGs and zero probability to the incompatible ones. Accordingly, the likelihood is approximated by

$$L(\Theta) \approx \frac{1}{M} \sum_{i=1}^{M} P(D|G_i; \Theta) \frac{P(G_i)}{Q(G_i)}, \tag{2.5}$$

where $G_1, G_2, ..., G_M \sim Q(G)$. For a good $Q(G)$, importance sampling is more efficient than (2.4). However, finding a good $Q(G)$ is challenging.

For the case of no recombination, Griffiths and Tavaré (1994) introduced a proposal that simulates compatible ARGs recursively. Soon after, Griffiths and Marjoram (1996) introduced the first importance sampling algorithm to perform inference in the presence of recombination. Their proposal distribution is a recursion for the probability of an ARG, and is derived by considering the next event in the ARG, going backward in time. Although it produced useful ideas for ARG inference, the algorithm is computationally extensive. Stephens and Donnelly (2000) introduced a more efficient proposal. They assumed that a haplotype could be constructed as an imperfect copy of the other haplotypes. Fearnhead and Donnelly (2001) extended Stephens and Donnelly's method to include recombination. They introduced a Hidden Markov Model (HMM) to approximate the likelihood by a set of conditional probabilities. Later, Li and Stephens (2003) adopted this idea and introduced an algorithm to approximate the likelihood.

2.5.1.1 Li and Stephens model

Li and Stephens (2003) argued that the coalescent is an approximation of reality under which inference is difficult, why not another approximation of reality

but with straightforward likelihood computation? If $D^1, D^2, ..., D^n$ are a sample of n haplotypes typed at m biallelic SNPs, they assumed that each of the haplotypes can be simulated from the allelic types of the rest of the haplotypes in the sample. With this view, the likelihood in equation (2.3) can be evaluated without taking on the expensive computation of an evolutionary process such as the coalescent.

In the Li and Stephens model, hereafter LS, the full likelihood is approximated by using a HMM to approximate the conditional probabilities,

$$P(D^1, ..., D^n|\rho) = P(D^1|\rho)P(D^2|D^1;\rho)...P(D^n|D^1, ..., D^{n-1};\rho). \tag{2.6}$$

The distribution of the first haplotype is assumed to be independent of ρ, so $P(D^1|\rho) = P(D^1) = 1/2^m$. The haplotype i (D^i) is simulated from the previous haplotypes, $D^j, 1 \leq j < i$, which are known as the *reference panel*. In other words, we compute the conditional probability $P(D^i|D^1, ..., D^{i-1};\rho)$. Also, let $C_l \in \{1, 2, ..., i-1\}$ denote which haplotype D^i copies at site l. Let D^j_l denote the allele at SNP l on haplotype D^j. To construct (or copy) the lth SNP of D^i, with

$$P(D^i_l|C_l = j, D^1, ..., D^{i-1}) = \begin{cases} \frac{i-1}{(i-1+\tilde{\theta})} + \frac{1}{2}\frac{\tilde{\theta}}{(i-1+\tilde{\theta})}, & D^i_l = D^j_l; \\ \frac{1}{2}\frac{\tilde{\theta}}{(i-1+\tilde{\theta})}, & D^i_l \neq D^j_l; \end{cases} \tag{2.7}$$

the D^i_l is either an exact copy or with a mismatch (mutation) from D^j_l, where

$$\tilde{\theta} = \left(\sum_{s=1}^{n-1} \frac{1}{s}\right)^{-1}.$$

Assume that D^i copies haplotype D^x at site l, i.e., $C_l = x$, at the next site, D^i can either copy D^x_{l+1} or jump to another haplotype. The transition probabilities are

$$P(C_{l+1} = x'|C_l = x) = \begin{cases} e^{(-\rho_l d_l/i-1)} + \frac{(1-e^{(-\rho_l d_l/i-1)})}{i-1}, & \text{if} \quad x' = x; \\ \frac{1-e^{(-\rho_l d_l/i-1)}}{i-1}, & \text{otherwise,} \end{cases} \tag{2.8}$$

where d_l is the genomic distance between sites l and $l+1$, and $\rho_l = 4Nr_l$, where r_l is the average rate of recombination per site between sites l and $l+1$. Using equations (2.7) and (2.8), the conditional probabilities in equation (2.6) are evaluated.

The LS has been used to answer a range of questions from estimating recombination rate to haplotype phase from haplotype data (Donnelly and Leslie, 2010). It changed the inference to a tractable problem, where evaluating the full likelihood is straightforward. However, it does not explicitly model past events and does not infer the ARG, limiting some inferences. In addition, the conditional probabilities in (2.6) depend on the order of the haplotypes and hence do not meet the feature of exchangeability of the true (unknown) distribution.

2.5.2 Reconstructing ARGs using heuristic methods

An oversimplification of the ARG inference problem is to find a point estimate rather than sampling from the posterior distribution. There are many heuristic algorithms in the literature that have been introduced to reconstruct one ARG from a given data. For instance see (Wu, 2009; Kececioglu and Gusfield, 1998; Song et al., 2005; Minichiello and Durbin, 2006; Nguyen et al., 2017; Hein, 1993). Most of these methods are computationally intensive (Rasmussen et al., 2014). Recently, two heuristic algorithms have been introduced to reconstruct an ARG for large data sets, which are presented in the following sections.

2.5.2.1 tsinfer

tsinfer, introduced by Kelleher et al. (2019), is a fast heuristic method that infers a single ARG consistent with a set of DNA sequences. The method assumes that the ancestral allele at each site is known, and mutations evolve according to the ISM. *tsinfer* leverages the features of TS and outputs a point estimate for the genealogical trees in the format of TS.

tsinfer begins with ordering sites by the frequency of the derived allele as a proxy for allele age. Then for each site, say j, the ancestral sequence is built by using the sequences that carry the derived allele at j, and the age of site j comparing to its neighboring sites.

Next, the genealogical relationship between these ancestors is found using a modification of LS (section 2.5.1.1) in which the reference panel for a given sequence is all the older ancestral sequences. Contemporary DNA sequences are considered imperfect mosaics of the recent ancestors, which are assumed to be mosaics of older ancestors. The copying process starts with the oldest ancestral sequence, and the corresponding partial TS is built for this sequence. Then, moving forward in time, the algorithm finds the most likely path for younger sequences using the Viterbi algorithm (Forney, 1973) from the available reference panel. The paths are encoded as TS edges and nodes (section 2.4.1.1), and the resulting output is a TS.

One of the desirable properties of *tsinfer* is its scalability. *tsinfer* can handle hundreds of thousands of genome-scale sequences. For instance, Kelleher and colleagues applied *tsinfer* to the UK Biobank data (Bycroft et al., 2018) with about 487×10^3 individuals (genome-wide). It took only about 3 hours to construct a TS on a server with 40 cores and 187 GB RAM.

The authors used the Kendall-Colijn (KC) metric (Kendall and Colijn, 2016), a tree distance metric, to assess the quality of the inferred topologies using a simulation study. For a given tree, the KC measures the distance from the MRCA of each pair of leaves and the tree root. The distance between two given trees is then compared by taking the Euclidian distance between the measured values of the trees. For a fixed mutation and recombination rate, the authors simulated 100 replicates of 16 sequences with 1 million sites using *msprime*. Then *tsinfer*, *rent+* (Mirzaei and Wu, 2017), *fastARG*[2], and *ARGweaver* (see section 2.5.3.1) are compared against each other using the KC metric. At

[2]https://github.com/lh3/fastARG

each site, the KC distance between the inferred tree and the truth is found. For an ARG, the KC is weighted by the length of the genomic interval spanned by each tree. *tsinfer* is reported to be more accurate than *rent+* and *fastARG*, and with similar accuracy to *ARGweaver*.

tsinfer does not output an ARG. It reconstructs the marginal trees in the form of the TS. Most of the information related to the recombination events, such as the number of recombination events and the ancestral sequence of recombinant lineages, are not inferred. Additionally, *tsinfer* does not infer the event times, but only the topologies of the marginal trees.

2.5.2.2 Relate

Relate is another heuristic ARG inference method developed by Speidel et al. (2019), which produces a single ARG consistent with the data. It assumes the ISM and that the ancestral alleles are known.

At each site, *Relate* uses a modification of the LS to prioritize the order of sequences to coalesce and builds a distance matrix whose rows contain information on the order of the coalescence event between one particular sequence and the others. This distance matrix serves as an input to a tree heuristic builder algorithm to build rooted binary trees at each site. After genealogical trees at each site are built, the coalescence times are estimated using an MCMC algorithm. Two changes are proposed to update the event ordering and event times, assuming the coalescent as prior. The first is to choose a coalescent event at random and swap it with another coalescent event consistent with the data. The second proposal is to update the time of an event according to an exponential distribution. The algorithm terminates as soon as each time interval has been updated at least 20 times.

To assess the accuracy of *Relate*, the authors compared the algorithm with *ARGweaver* in estimating the pairwise TMRCA at each site using simulated data. Figure 2.5 compares the true TMRCA of pairs of sequences to the inferred ones by *Relate* and *ARGweaver* for a simulated data of 50 sequences with 2.5 million sites. For *ARGweaver*, the pairwise TMRCA at each site is an average over all the retained ARGs. It is seen that *Relate* infers the pairwise TMRCAs accurately. However, *ARGweaver* tends to underestimate the more recent coalescence events.

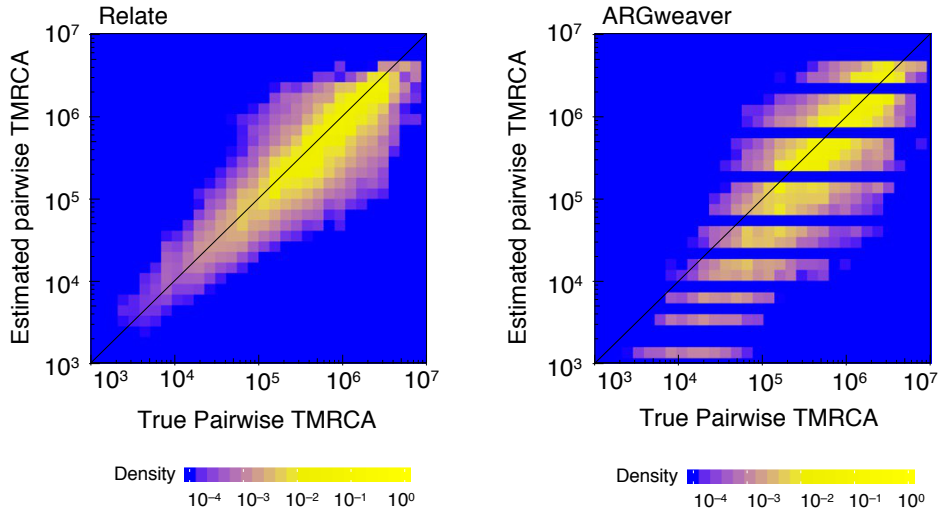


Figure 2.5: The true TMRCA for pairs of sequences versus those inferred by *Relate* and *ARGweaver*. This figure is taken from Speidel et al. (2019).

Similarly to *tsinfer*, a powerful property of *Relate* is its scalability; it handles thousands of sequences genome-wide. Moreover, *Relate* infers both event times and the topologies, whereas *tsinfer* only infers the topologies.

Relate and *tsinfer* provide only point estimates for the ARG and lack measures of uncertainty. The constructed ARG can be any of the infinitely many possible compatible ARGs, and no statement can be made about the accuracy of the estimates. A more challenging approach is to develop probabilistic models that allow for ARG uncertainty. Bayesian models, such as MCMC, can approximate the posterior distribution and provide additional information about parameters. However, this extra information comes at a high computational cost.

2.5.3 Markov Chain Monte Carlo

We attempt to infer full distributions rather than point estimates. A Metropolis-Hastings (MH) algorithm (Hastings, 1970; Metropolis et al., 1953) can be designed to produce dependent samples from $P(G|D;\Theta)$, i.e., the CwR+. The algorithm starts with setting initial values for the parameters (Θ) and the ARG (G_j). Then, a new ARG (G_{j+1}) is proposed using a distribution $Q(.)$, which may depend on the current ARG. The proposed ARG is accepted with probability

$$A = \min\{1, \frac{P(D|G_{j+1};\Theta)P(G_{j+1};\Theta)}{P(D|G_j;\Theta)P(G_j;\Theta)} \times \frac{Q(G_j|G_{j+1})}{Q(G_{j+1}|G_j)}\}; \tag{2.9}$$

otherwise, G_j is kept. Given that this process is repeated long enough, a series of ARGs visited by the Markov chain represent a series of dependent samples from $P(G|D;\Theta)$ (Wakeley, 2009).

It is common to discard the first $h \geq 0$ samples from the Markov chain, called *burn-in*. Since MCMC algorithms are problem-specific and convergence

speed can differ between them, there is no fixed rule on how large should h be or when to terminate the algorithm. The MCMC samples after the *burn-in* period can be used to estimate the parameters of interest. These samples may be highly correlated. It is useful to retain every kth sample to reduce the autocorrelation of the samples. k can be found so that the autocorrelation of the samples is lower than a certain threshold.

The last term in equation (2.9) is the Hastings term (Hastings, 1970). These conditional probabilities are named *transition probabilities* under the proposal distribution. $Q(G_j|G_{j+1})$ (*the reverse transition probability*) is the probability of proposing G_j if the current state is G_{j+1}. Similarly, $Q(G_{j+1}|G_j)$ (*forward transition probability*) is the probability of proposing G_{j+1} if the current state is G_j.

An early effort to use an MCMC approach for inferring the coalescent without recombination was made by Kuhner et al. (1995) and soon after modified by Felsenstein et al. (1999); at each MCMC iteration, a branch is randomly chosen and rejoins an older existing lineage (backward in time). Kuhner et al. (2000) extended the model to include recombination under a finite sites model of mutation. Their proposal distribution is discussed in section 5.5.6. Some other MCMC algorithms including Nielsen (2000) (under the ISM) and Wilson and Balding (1998) (in the absence of recombination) have been proposed. Although they produced useful ideas for ARG inference, the algorithms scaled poorly both with sample size and sequence length. For a review of the performance of these methods, refer to Wall (2000).

In the following sections, we present two of the more recent MCMC algorithms.

2.5.3.1 ARGweaver

ARGweaver, introduced by Rasmussen et al. (2014), is a probabilistic model and a substantial advance over other methods in terms of accuracy. *ARGweaver* assumes a discretized approximation of SMC (DSMC) and samples from the posterior distribution, which we call DSMC+.

ARGweaver employs some simplifying assumptions to decrease computational complexity. Firstly, it works under a time-discretized version of the SMC where both event times and chromosome sites are assumed to be discrete. In other words, all the events, i.e., recombination and coalescent events, are assumed to occur on a set of (user-specified) discrete time points. By default, the time points are placed uniformly on a logarithmic scale so that recent times are discretized into shorter intervals than older times. Secondly, only one recombination event is allowed to occur between two adjacent sites. As a result, two neighboring distinct trees differ only by a single recombination event, making the Subtree-Pruning-and-Regrafting (SPR) operation easier to perform. Although the probability of having more than one recombination is low in general, this simplification may lead to biased estimation when the recombination rate is high, and the sample size or the effective population size is large.

Furthermore, the effective population size, the mutation rate, and the recombination rate are known and fixed. *ARGweaver* allows for variable rates

across the chromosome and variable N for time intervals. However, all must be specified by the user.

For modeling the mutation process, *ARGweaver* uses the Jukes-Cantor model (Jukes et al. (1969)) that assumes that nucleotide substitutions occur at an equal rate. With the Jukes-Cantor assumption, D is obtainable from any given ARG. Therefore there is no incompatibility between the data and the proposed ARG, which is a major problem in proposing an ARG under the ISM (see section 4.5.3).

For a sample of DNA sequences (D), and a set of parameters, $\Theta = (N, \mu, r)$, *ARGweaver* reformulates the DSMC as a HMM and uses an MCMC algorithm to sample from the DSMC+. Here, an ARG is a set of marginal trees (T^n) and the information about the recombination times and breakpoints (R^n), i.e., $G^n = (T^n, R^n)$. The basic idea behind this method is to remove a sequence from an ARG of n sequences G^n, resulting in a floating sequence and an ARG of $n-1$ sequences, G^{n-1}. Then, "thread" the floating sequence back to the G^{n-1} by resampling all its coalescent points in each marginal tree and all recombination events, if applicable, under the DSMC assumption. A HMM model is used to optimize the path to offer this threading operation and build a new G^n.

One problem is that rearranging the leaves of the trees does not necessarily result in an efficient rearrangement of the internal branches. Notably, for large datasets, threading a leaf sequence can rarely rearrange the internal branches up in the ARG near the GMRCA, which leads to poor mixing. The authors proposed an alternative to "thread" not only the sequences but also the internal branches, called "subtree threading." In this method, removing an internal branch divides the tree into a subtree and a main tree, and then using the threading operation, the subtree rejoins to the main tree.

Hubisz (2019) implemented the SMC' in *ARGweaver* and showed that the inference quality is slightly improved for recombination rates. However, no significant improvement is observed over the original version for most ARG parameters, such as the TMRCA and the ARG total branch length. Also, the algorithm is slower.

Although *ARGweaver* cannot handle the currently available large data sets, it can handle a few tens of sequences. For a sample of 20 chromosomes with 10^6 sites, *ARGweaver* requires about 11 hours to run 5×10^3 MCMC iterations (on a server with 1 core). In chapter 6, we compare our method against *ARGweaver* and discuss its strengths and weaknesses.

2.5.3.2 Arbores

More recently, another MCMC algorithm (*Arbores*) has been introduced by Heine et al. (2018) for inferring the ARG under the SMC and the ISM assumptions. *Arbores* explores the ARG space by re-simulating the genealogical trees, called "bridges", of a segment of genome, conditioning on the trees of the first and last sites.

At each MCMC iteration, a genomic interval, $[a, b]$, is chosen. Moving rightward from a, all the possible bridges for the interval are simulated by applying all possible SPR moves on the trees. Next, the event times are generated for the

new nodes. The bridges that disagree with the trees at sites a and b or are inconsistent with the ISM are discarded. Lastly, one bridge is randomly chosen among the remaining bridges and is accepted (or rejected) using an MH criterion.

Although the bridges are simulated in parallel, *Arbores* is slow and not applicable to more than a handful of sequences with length about 10^4 sites. The authors applied the algorithm to the Kreitman data (Kreitman, 1983) with 9 sequences comprising 2287 sites (30 SNPs). It took 10 hours to run 2×10^5 MCMC iterations. We argue that one of the reasons for this poor performance can be the uniform sampling of bridges. Many of the generated bridges are incompatible with the data or with the initial and terminal trees. Therefore, the algorithm spends a large amount of time producing incompatible paths.

The authors applied some heuristics to speed up the algorithm. For instance, they limited the number of SPR operations between two SNPs to be at most one. Also, the number of recombinations is restricted so that a new recombination event is introduced only if it is required in the data. Even with these heuristics, no significant improvement is reported.

The authors compared their model (including heuristics) against *ARGweaver* on a simulated data consisting of eight sequences across 10^4 sites. Figure 2.6 shows the trace plot of the number of recombinations for *Arbores* and *ARGweaver*. It seems that *ARGweaver* mixes better. The estimated number of recombinations for both algorithms is between 5 and 6.

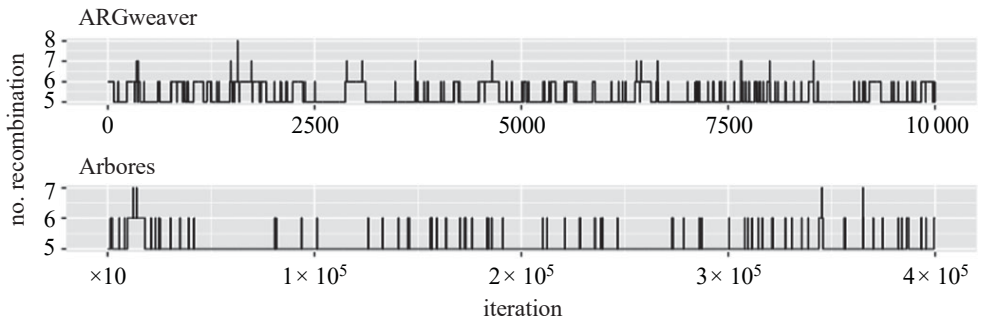


Figure 2.6: The trace plot of the number of recombination for *ARGweaver* and *Arbores*. This Figure is taken from Heine et al. (2018).

In conclusion, *Arbores* does not improve the quality of inference over *ARGweaver* and handles a smaller number of sequences.

2.6 Discussion

In this chapter, we discussed the coalescent with and without recombination and its approximations. We also explained a recent breakthrough, the TS, that efficiently simulates and stores millions of sequences under the CwR.

The likelihood function is not closed form, so we require integrating over all the possible ARGs to calculate the likelihood. Importance sampling suggests sampling from a proposal distribution rather than the CwR. With a good

proposal distribution, importance sampling can reduce the complexity of the likelihood evaluation. However, finding a good proposal distribution is challenging, and the introduced proposals are computationally expensive. We then explained some heuristic methods that construct a single ARG for the given data. *tsinfer* and *Relate* are such methods and can handle large data sets. However, they do not quantify uncertainty.

A more accurate but expensive method is MCMC that approximates the posterior distribution and provides uncertainty measures for all the parameters. *ARGweaver* uses this approach to infer ARG under DSMC. The SMC does not model TNAM, and consequently, no non-ancestral recombination exists in the model. However, this information contributes to the genetic structure of the sample (see section 2.4.1). Section 3.2 shows that the amount of TNAM grows rapidly with sequence lengths so that non-ancestral recombination events outnumber ancestral recombinations for sequence lengths above about 2.5×10^5 sites in human. Ignoring this large amount of information influences the accuracy of parameter estimations, for instance, the recombination rate.

Inference under the CwR remains a challenge in population genetics. The TS has gained considerable attention for its efficiency gains in simulating and storing large DNA sequences. There is a point estimator (*tsinfer*) for the ARG that uses TS and handles large available data sets. However, no attempt has been made to perform probabilistic inference under the CwR by taking advantage of the TS. My goal is to be the first to attempt this and investigate whether efficiency gains can be obtained.

Chapter 3

Deficiencies in current methods

In this chapter, I discuss two motivations that inspired me to tackle the CwR inference problem. First, the existing probabilistic inference models do not exploit the idea of TS in representing ARGs. In section 3.1, I show why it is important to do so. Second, in section 3.2, I show that the TNAMs form a large portion of an ARG and discuss its importance.

3.1 Tree Sequence data structure

Representing an ARG so that the embedded information is easily accessible and redundancies are avoided is not straightforward. The genealogical trees present in an ARG share subtree with their neighbors. To illustrate, Figure 3.1 shows an ARG of three sequences, each with length 10 sites. Panel *(b)* presents the marginal trees for genomic intervals $[1, 2]$, $[3, 7]$, and $[8, 10]$, respectively. As seen from the Figure, the subtree below node d is identical in all three trees. The common format, particularly in SMC based methods, is to build the marginal trees separately so that identical subtrees are stored for each tree, which is inefficient.

As discussed in section 2.4.1.1, Kelleher et al. (2016) introduced the TS to represent marginal trees so that the identical subtrees of neighboring trees are stored only once. This idea leads to a significant saving in storage and speed in processing and accessing ARG information (Kelleher et al., 2016). Our goal is to employ this efficiency in inferring the CwR to improve the speed and accuracy.

To better understand the similarity of adjacent trees, we conducted a simulation study. We simulated 500 ARGs under the CwR using *msprime* with the settings $\mu = r = 10^{-8}$ and $N = 10^4$ for various n and L, where n is the sample size and L is the sequence length (see Table B.1).

Recall that for a random sample of n DNA sequences, the number of branches for a coalescent tree at a non-recombining segment of the genome is $2n-2$. Let us denote the number of distinct recombination breakpoints along the genome as N_b. Therefore, the total number of tree branches of an ARG is

$$T_b = (2n - 2)(N_b + 1).$$

Table 3.1 represents the proportion of branches shared between consecutive trees in the history of $20, 100$, and 500 sequences with length $10^5, 5 \times 10^5, 10^6$,

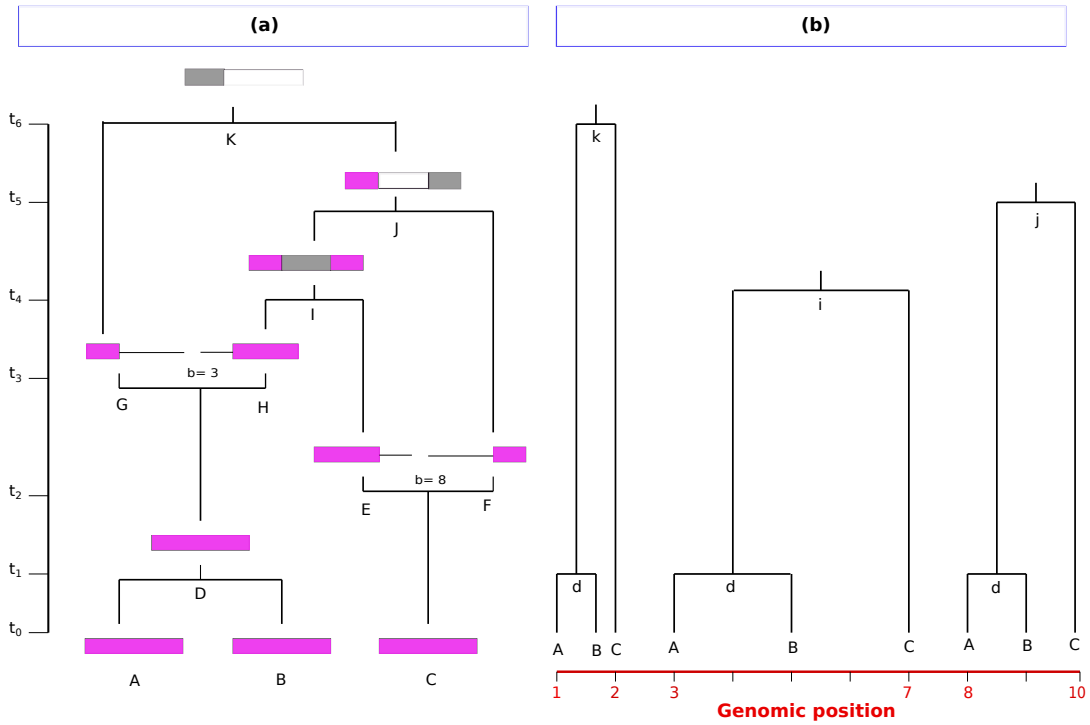


Figure 3.1: *(a)*: An ARG of three sequences with length ten sites. The purple color represents material ancestral to the samples; the grey color indicates that the MRCA has just been reached. No color (white) on the lineages J and K indicates that the MRCA has already been reached for the interval and are treated as non-ancestral material. *(b)*: The corresponding marginal trees.

n	L	$N_t = 1$	$2 \leq N_t < 10$	$10 \leq N_t < 20$	$20 \leq N_t < 50$	$50 \leq N_t$
20	100K	0.0247	0.1915	0.1478	0.2441	0.3919
	500K	0.0238	0.1776	0.1337	0.2119	0.4530
	1M	0.0236	0.1765	0.1322	0.2075	0.4602
	2M	0.0236	0.1758	0.1317	0.2067	0.4622
100	100K	0.0026	0.0351	0.0444	0.1210	0.7969
	500K	0.0024	0.0315	0.0375	0.0928	0.8358
	1M	0.0023	0.0311	0.0367	0.0902	0.8397
	2M	0.0024	0.0309	0.0361	0.0879	0.8427
500	100K	0.0005	0.0053	0.0084	0.0292	0.9566
	500K	0.0004	0.0046	0.0067	0.0201	0.9682
	1M	0.0005	0.0045	0.0065	0.0189	0.9696
	2M	0.0005	0.0045	0.0063	0.0184	0.9703

Table 3.1: The proportion of branches shared by N_t consecutive trees in the history of n DNA sequences with length L. Each point is an average of 500 ARGs simulated using *msprime*. $N_t = 1$ represents the proportion of unique branches, i.e., branches which exist only in one tree.

and 2×10^6 sites. As we observe, the proportion of branches shared by only one tree, $N_t = 1$, is below about 2.5% for all the settings.

As seen from Table 3.1, over 97.5% of the branches are shared by at least two neighboring trees. The proportion of the branches shared by at least 50 consecutive trees, $N_t \geq 50$, is over $0.40, 0.80$, and 0.95 for $n = 20, 100$, and 500, respectively. This high level of similarity among the neighboring trees highlights the importance of considering identical subtrees in the data structure.

3.2 Trapped non-ancestral materials

As discussed in section 2.4.1, aside from ancestral material, the non-ancestral materials trapped between two ancestral material segments, i.e., TNAM, affect the genetic structure of the data; the flanking ancestral material is carried by the same ancestor, which impacts Linkage Disequilibrium (LD). These materials are not modeled in the SMC but are present in the CwR. To assess the impact of TNAM, we first investigate the amount of TNAM present in an ARG. Secondly, the frequency of non-Markovian patterns in the TMRCA along the genome is examined.

3.2.1 Amount of TNAM in an ARG

We simulated ARGs under the CwR for 100 replicates of varying sequence length and sample size. Figure 3.2 shows the number of total, ancestral, and non-ancestral recombination events of the simulated ARGs. The number of non-ancestral recombinations is proportional to the amount of TNAM in an ARG. In the left panel, the sample size is fixed at 50, and the sequence length varies. As seen from the Figure, the number of non-ancestral recombination

events increases rapidly with sequence length so that it intersects the ancestral recombination line at $L \approx 2.5 \times 10^5$ sites and approaches the total number of recombination events. The number of SNPs has a power-law and linear relationship with the number of non-ancestral and ancestral recombinations, respectively. In the right panel, we fix $L = 2 \times 10^6$ sites and let the sample size vary. It is seen that the majority of the total recombination events are non-ancestral. SMC based methods such as *ARGweaver* only infer the ancestral recombination events, which are linear (green line), and do not capture the remaining. However, the high amount of TNAM suggests that the ancestral sequences and their sequence length might considerably differ from the SMC.

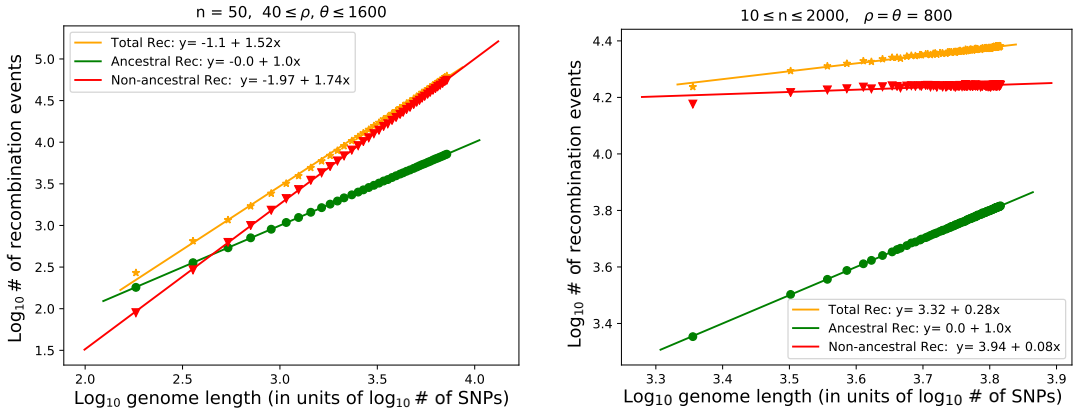


Figure 3.2: The logarithm of the mean number of total (yellow), ancestral (green), and non-ancestral (red) recombination events in simulated ARGs under the CwR over 100 replicates of varying sequence length (or ρ) and sample size. The lines are the fitted regression line. In the left panel we fix $n = 50$ and let the sequence length vary in the range of $(1 \times 10^5, 4 \times 10^6)$ sites. In the right panel we fix $L = 2 \times 10^6$ sites ($\rho = 4NrL = 800$), and vary the sample size from 10 to 2000.

3.2.2 Non-Markovian behaviour in the ARGs

McVean and Cardin (2005) quantified the importance of CA events between lineages with no overlapping ancestral material for two sequences. As defined in McVean and Cardin (2005), let us denote $Q^*(n, \rho)$ as the probability that n sequences that share a TMRCA at points 0 and 1 (continuous genome with the unit interval) do not share one in at least one intervening interval, given the recombination rate ρ. These CA events generate the non-Markovian property in the ARG; therefore, $Q^*(.)$ quantifies the amount of non-Markovian behavior in the CwR. Note that under the SMC, $Q^*(n, \rho) = 0$; hence, the value under the CwR can be interpreted as a measure of the error in SMC, or equivalently the importance of TNAM.

Using Monte Carlo simulation, McVean and Cardin (2005) obtained $Q^*(2, \rho)$ for various ρ, and reported that the strongest non-Markovian behavior occurs for $\rho = 10$, where $Q^*(2, 10) = 0.123$. From 10^6 replicates of simulated ARGs, they reported that only 11% shared the TMRCA for points 0 and 1. Consequently, they concluded that the probability of a non-Markovian event is less than 2%.

We extended their simulation study to investigate the non-Markovian behavior for $n \geq 2$. We simulated 10^7 replicates of ARGs under the CwR for varying n and ρ using *msprime*. Table 3.2 presents the obtained probabilities using Monte Carlo simulation for $2, 5, 20, 50$, and 100 sequences with various recombination rates. For $n = 2$, the results are consistent with the reported results in McVean and Cardin (2005). However, as seen from Table 3.2, $Q^*(n, \rho)$ is significantly higher for larger number of sequences. We observe that $Q^*(n, \rho) \approx 1$ for most of them, indicating that non-Markovian behavior is an important property in ARGs with more sequences.

Table 3.2: $Q^*(n, \rho)$. Each value is an average over 10^7 replicates of simulated ARGs using *msprime* with $r = \mu = 10^{-8}$, and $N = 10^4$.

ρ	4	40	200	400
$Q^*(2, \rho)$	0.094	0.079	0.023	0.012
$Q^*(5, \rho)$	0.406	0.992	1	1
$Q^*(20, \rho)$	0.514	1	1	1
$Q^*(50, \rho)$	0.520	1	1	1
$Q^*(100, \rho)$	0.521	1	1	1

McVean and Cardin (2005) also investigated the probability that the sequences share a single TMRCA at the beginning and endpoints. This probability is low and is decreasing with increasing sample size and sequence length. The authors concluded that because these genealogies are rare, the effect of non-Markovian behavior is negligible in general. On the other hand, the effect of non-Markovian behavior is substantial. For $n > 2$, $Q^*(n, \rho)$ is not able to capture all the effects of non-Markovian behavior because, firstly, there might be some non-Markovian events that do not affect the TMRCA of the endpoints such as invisible non-ancestral recombination events. Secondly, $Q^*(n, \rho)$ only contains one class of non-Markovian genealogies, where TMRCA of the first and last site is the same, and filters out all other classes. For instance, for $0 < x, z < 1$ with equal TMRCA, there might be at least one intervening point $x < y < z$ with a different TMRCA.

3.3 Discussion

In this chapter, we showed that there is a high level of similarity between marginal trees in an ARG. Taking into account these similarities has been proven to help gain simulation efficiency. In this thesis, I investigate their impact on the ARG inference problem.

We also discussed an important property of ARGs that has been neglected before. We showed that an ARG consists of a large amount of TNAM that affects the correlation structure of the data. We expect that modeling TNAMs will improve the quality of inference, particularly for the recombination rate.

In the next chapter, I introduce a probabilistic method that takes advantage of TS and models TNAMs.

Chapter 4

An MCMC algorithm for the CwR exploiting the TS

This chapter includes an initial attempt at an MCMC algorithm that did not succeed well. We assess its limitations and use this to develop a better algorithm in the following chapter.

In section 4.1, we discuss that the *TS* does not contain all the information required for inferring the CwR. Therefore, a new data structure (called *FTS*) is introduced to represent the full ARG so that the benefits of the TS are preserved. *FTS* consists of five tables which are explained in section 4.2. In section 4.3, we introduce an MCMC approach to sample from the CwR+. Section 4.3.1 details an algorithm to build an initial ARG under the *FTS* format from a sample of DNA sequences. In section 4.3.2, we develop methods to evaluate the likelihood and the prior under the new data structure. In section 4.3.3, some transitions are introduced to explore the space of the ARG. The performance of the algorithm is assessed using simulation data in section 4.4. We conclude that the algorithm performs poorly in inferring ARG properties. Section 4.5 reviews the limitations and the complexities of the introduced data structure as the potential reasons for the poor performance.

4.1 TS does not include all the ARG information

TS does not contain all the information of an ARG, but only the marginal trees. To be able to perform inference under the CwR, the following information is needed:

- Recombination and non-overlapping CA events.
- Recombination breakpoints and times.
- The parent ancestral sequences of recombinations.

This information is needed to be able to evaluate the prior and to explore the ARG space. More importantly, the TS cannot uniquely define an ARG. Infinitely many ARGs can be found for a single TS depending on the number of CA, invisible and double-hit (multiple recombinations at the same site) recombinations, and the time of recombination events.

4.2 Full TS data structure

We extend the TS to record the required information in a new data structure, called *Full TS (FTS)*, consisting of five linked tables. Before describing the tables, it is necessary to discuss a few points. We distinguish between an "ARG branch" and a "tree branch." An ARG branch resulting from a coalescence event contains all ancestral segments of the child branches. However, a tree branch only includes the shared ancestral segments. For instance, in Figure 4.1, a coalescence event occurs between branches H and E, which forms an ARG branch I and a tree branch i in the marginal trees. In general, a tree branch is a subset of the corresponding ARG branch. Here, the ARG branch I and tree branch i carry ancestral material for segments $[1, 10]$ and $[3, 7]$, respectively.

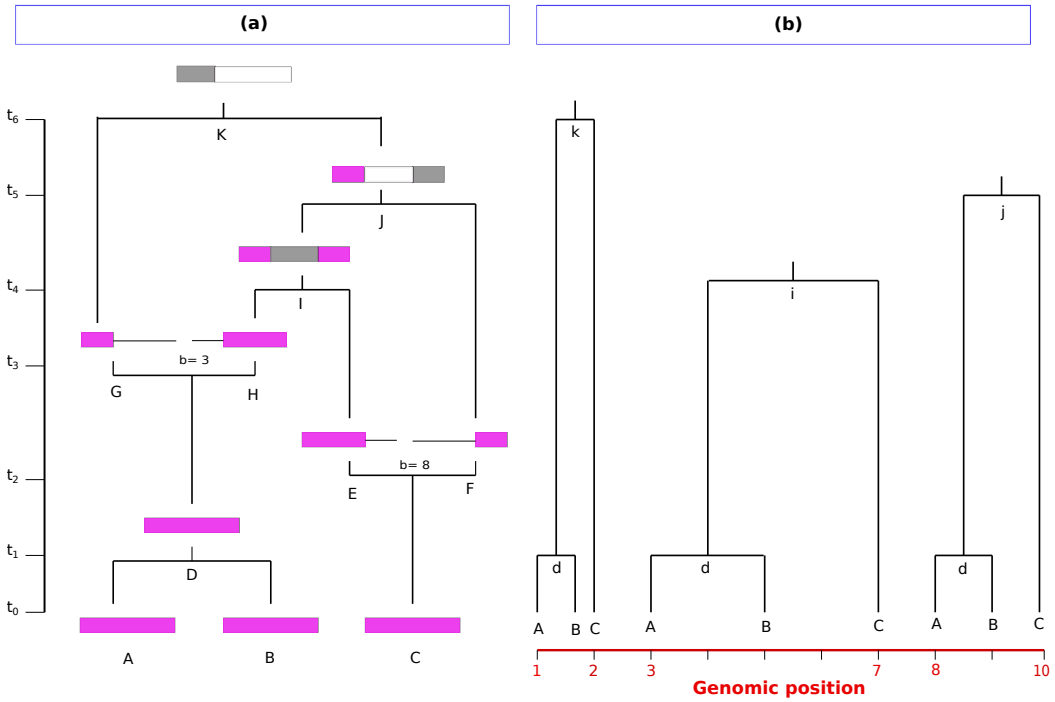


Figure 4.1: This Figure is a copy of Figure 3.1. *(a)*: An ARG of three sequences with length ten sites. The purple color represents the ancestral material for the sequences; the grey color indicates that the MRCA has been reached for the interval. No color (white) on the lineages shows that the MRCA been already reached for the interval. *(b)*: The corresponding marginal trees.

4.2.1 Table 1: Event List

The ancestral events and related information are recorded in the Event List (EL) Table. For each event, the event type (either recombination "0" or CA "1"), the event time, the ARG parent and child branches, the difference between the total recombination links after and before the event, and the recombination breakpoint (if the event is recombination) is stored. For example, the first event of the ARG (looking backward in time) shown in Figure 4.1 is a CA event (coalescence event in particular) between A and B, resulting in ARG branch D,

and is recorded as $[1, D, (A, B), t_1, -9]$, where -9 is the difference between the parent and child recombination links $(9 - 18)$ (Figure 4.2). This table contains enough information to evaluate the prior.

EL

e	P	ch	t	diff	b
1	D	[A, B]	t_1	-9	-
0	[E, F]	C	t_2	-1	8
0	[G, H]	D	t_3	-1	3
1	I	[H, E]	t_4	-4	-
1	J	[I, F]	t_5	-9	-
1	K	[J, G]	t_6	-2	-

ASC1

AN	l	r	TN
A	1	10	A
B	1	10	B
C	1	10	C
D	1	10	d
E	1	7	C
F	8	10	C
G	1	2	d
H	3	10	d
I	1	2	C
I	8	10	d
J	1	2	C

ASC2

AN	t_s	t_e	link
A	t_0	t_1	9
B	t_0	t_1	9
C	t_0	t_2	9
D	t_1	t_3	9
E	t_2	t_4	6
F	t_2	t_5	2
G	t_3	t_6	1
H	t_3	t_4	7
I	t_4	t_5	9
J	t_5	t_6	1

CR

l	r	P	CH	t
1	10	d	[A, B]	t_1
3	7	i	[d, C]	t_4
8	10	j	[d, C]	t_5
1	2	k	[d, C]	t_6

MRCA

AN	l	r	TN	t
I	3	7	i	t_4
J	8	10	j	t_5
K	1	2	k	t_6

Figure 4.2: The *FTS* tables to present the ARG shown in Figure 4.1. For details see the text.

4.2.2 Table 2: Coalescence Records (CR)

This table records the marginal trees. When a coalescence event occurs, the overlapping genomic intervals, the parent tree branch, the child tree branches, and the event time are stored in this table. For instance, the event at time t_1 on the ARG in Figure 4.1 is a coalescence event. As seen from Figure 4.2, the recorded information is $[1, 10, d, (A, B), t_1]$, where 1 and 10 are the left and right sides of the merged ancestral material. We use this table to evaluate the likelihood.

4.2.3 Table 3: Ancestral Segment Composition (ASC)

The ancestral regions on the ARG branches are stored in this table. To preserve a connection between the ARG branches and tree branches, we also store the corresponding tree branches. This information is needed in the MCMC algorithm to be able to rearrange the ARG topology. An ARG branch may have more than one entry if there is more than one corresponding tree branch. To illustrate, branch I in Figure 4.1 contains ancestral material in the intervals $[1, 2]$ and $[8, 10]$. From panel (b), we can see that the tree branches for these intervals are C and d, respectively. Therefore the ASC, as shown in Figure 4.2, records two arrays $[[I, 1, 2, C], [I, 8, 10, d]]$ for this event.

4.2.4 Table 4: ARG Branches (AB)

Aside from the information recorded in the ASC, the initiation and termination time and the number of recombination links are required to specify an ARG branch. Since there might be multiple entries per ARG branch in ASC, to avoid repetition, here we separately record this information. The initiation time for a branch is when they appear. The termination time is when the branch experiences an event. For example, ARG node E in Figure 4.1, the initiation time (t_2), the termination time (t_4), and the number of recombination links ($7-1$) are recorded in AB.

4.2.5 Table 5: MRCA table

The genomic intervals that have attained their MRCA are stored in this table. This information also is needed in the update step of the MCMC algorithm. Aside from the genomic intervals, the time of the events and the corresponding tree branches are recorded. The interval $[8, 10]$ in the ARG branch J has attained its MRCA (Figure 4.1); accordingly, $[J, 8, 10, j, t_5]$ is preserved in the MRCA table, where j is the corresponding tree branch and t_5 is the event time.

Table 4.1 presents the notation used in the reminder of the chapter.

Table 4.1: Table of notation for chapter 4, excluding the symbols in Table B.1.

Initial ARG (section 4.3.1):	
B	a set of lineages among the existing lineages that have zero mismatches with lineage u
P_C	probability of a CA event
t	time of the current state of the algorithm
Computing probabilities (section 4.3.2):	
c_i, d_i	the child branches of the ith row of the CR table
$\lvert m_{c_i}\rvert$	the number of mutations on branch c_i
l_{c_i}	the length of branch c_i
l_i, r_i	the left and right end sites of genomic region in the row i of CR table
$\lvert CR\rvert$	the number of rows in CR table
Proposal distribution (section 4.3.3):	
d	the pruned branch in the SPR
F	a set of floating lineages
t_u	the initiation time of branch u
k_i	the number of active lineages in (t_i, t_{i+1})
$P(f)$	incomplete forward transition probability

4.3 The MCMC approach

We assume that the effective population size (N), the mutation rate per site per generation (μ), and the recombination rate per site per generation (r) are known. In addition, the ancestral alleles for all of the segregating sites are

given, and the CwR is assumed. The mutation model is an approximation of ISM, where the chromosome is assumed to be discrete (see section 2.2). Under the ISM, an ARG is called *incompatible* with D if it cannot generate D with exactly one mutation per segregating site.

The MCMC algorithm has three steps, which are presented in the following.

4.3.1 MCMC step 1: Construct an initial ARG

Given that Θ is known, we need to set an initial ARG for D. Finding an ARG compatible with D can be challenging. We borrowed some ideas from Nguyen et al. (2017) and devised a heuristic algorithm to construct a compatible ARG from $D = \{D^1, D^2, ..., D^n\}$. Let 0 and 1 denote the ancestral and derived alleles at a site, while -1 indicates that the ancestral state is unknown. The algorithm has three main operations:

- ***Mutation operation:*** let D_i^j denote the allele at the ith site of D^j. If for sequence i, $D_i^j = 1$, but for all other sequences $D_k^j = 0$ ($k \neq i$), set D_i^j to 0.
- ***Recombination operation:*** select a lineage (u) among all the available lineages proportional to their number of recombination links. If the number of recombination links on u is more than 1, a genomic position (x) is chosen uniformly. u is split into two lineages, one of which carries the ancestral material to the left of x, and the other carries the ancestral material to the right of x.
- ***Coalescent operation:*** a lineage (u) is randomly selected from the current lineages. Let B be the subset of the current lineages that match u at all shared ancestral sites. If $B \neq \phi$, choose a lineage ($v \in B$) that has the highest number of overlapping genomic sites with u. Coalesce u and v.

The steps of the algorithm are as follows:

1. Set $k = n$, $k' = n(L-1)$, and $t = 0$, where k' is the total number of recombination links and t is the time.
2. Draw a time point (t') from an exponential distribution with rate
$$\lambda = \frac{\binom{k}{2}}{2N} + rk'.$$
3. Apply ***Mutation operation*** to D.
4. With probability
$$P_C = \frac{\binom{k}{2}/2N}{\lambda},$$
the new event at time $t + t'$ is a CA (go to step 5). Otherwise, go to step 6.
5. If a CA event occurs, apply ***Coalescent operation***.

- If $B \neq \phi$,
  - if the parent node is a root (see Definition 4.3.4), then $k = k - 2$, otherwise $k = k - 1$. Go to step 2.
- If $B = \phi$, go to step 3.

6. If a recombination event occurs, apply ***Recombination operation***. If the number of links on v is greater than one, then update $k = k + 1$ and go to step 2. Otherwise, go to step 3.

7. Continue until $k = 1$.

We store the output of this algorithm in the form of *FTS*.

4.3.2 MCMC step 2: Compute the likelihood and prior

4.3.2.1 Likelihood evaluation

Given an ARG, if b is a branch of a marginal tree in the ARG, the probability of m_b mutations on b has a poisson distribution

$$P(m_b) = \frac{(l_b\mu)^{m_b}}{m_b!}(e^{-l_b\mu})^{S_b}, \tag{4.1}$$

where l_b is the branch length and S_b is the tree interval, i.e., the length of the genomic interval of the tree. Applying equation (4.1) to all of the branches of the marginal trees on the ARG leads to the likelihood,

$$P(D|G,\Theta) = \prod_b \frac{(l_b\mu)^{m_b}(e^{-l_b\mu})^{S_b}}{m_b!}. \tag{4.2}$$

We introduce an algorithm to evaluate the likelihood using the *CR* table. Let $(l_i, r_i, p_i, (c_i, d_i), t_i)$ denote the left genomic side, the right genomic side, the parent branch, the first child, the second child, and the time of the ith row of the *CR*, respectively, where $i = 1, 2, ..., |CR|$, and $|CR|$ is the number of rows. The rows are ordered so that $t_i < t_{i+1}$. The algorithm is as follows:

1. Set $i = 1$ and $P = 1$.

2. Find the number of mutations $|m_{c_i}|$ and $|m_{d_i}|$ on child branches c_i and d_i by comparing the SNPs on c_i and d_i against each other from site l_i to r_i.

3. Let l_{c_i} and l_{d_i} denote the branch lengths of c_i and d_i, respectively. Calculate the probability of $|m_{c_i}|$ and $|m_{d_i}|$ mutations on c_i and d_i by

$$A = (e^{-\mu l_{c_i}})^{r_i - l_i}(\mu l_{c_i})^{|m_{c_i}|}(e^{-\mu l_{d_i}})^{r_i - l_i}(\mu l_{d_i})^{|m_{d_i}|}.$$

4. Multiply P by A.

5. Increment i by one ($i = i + 1$) and go to step 2.

6. Continue until $i = |CR|$.

The resulting

$$P = \prod_{i=1}^{|CR|} (e^{-\mu l_{c_i}})^{r_i - l_i} (\mu l_{c_i})^{|m_{c_i}|} (e^{-\mu l_{d_i}})^{r_i - l_i} (\mu l_{d_i})^{|m_{d_i}|}, \tag{4.3}$$

is the likelihood $P(D|G, \Theta)$. Note that the algorithm visits the identical subtrees of neighboring trees only once. Since the ratio of the likelihoods is used in the MH term of the MCMC algorithm, the denominator of (4.2) cancels out, and we can use (4.3).

4.3.2.2 Prior evaluation

For an ARG with E events with times $t_1, ..., t_E$ generations, we evaluate the prior by considering all the events and calculating the probability of that certain event at that time.

Assume that k_i and k'_i are the total number of lineages and recombination links immediately before time t_i, respectively. Note that we move backward in time, so "before" is closer to the present. Under the CwR, the time to the next event is exponentially distributed with rate

$$\lambda_i = \frac{k_i(k_i - 1)}{4N} + rk'_i. \tag{4.4}$$

The event at t_i is a CA with probability

$$\frac{e^{-\lambda_i t_i}}{2N}, \tag{4.5}$$

or a recombination with probability

$$re^{-\lambda_i t_i}. \tag{4.6}$$

Thus, the prior is calculated by

$$\prod_{i=1}^{E} [I_c \frac{e^{-\lambda_i t_i}}{2N} + (1 - I_c) re^{-\lambda_i t_i}], \tag{4.7}$$

where I_c is an indicator function with value 1 if the event is a CA, and 0 otherwise.

The *EL* table contains all the information needed to evaluate the prior.

4.3.3 MCMC step 3: Update the ARG

Let $X(j)$ denote the state of the MCMC after j iterations, and the ARG at $X(j)$ is G_j. To explore the ARG space, we propose changes on the current ARG according to a proposal distribution $Q(.)$. The transition types in $Q(.)$ are as follows:

1. Perform a local Subtree-Pruning-and-Regrafting (SPR) operation on the ARG.

2. Remove an existing recombination event at random.
3. Add a recombination event to the ARG.
4. Re-sample the event times.
5. Modify the breakpoint of an existing recombination.
6. Exchange the order of two consecutive events that are not comparable.

Applying $Q(.)$ to G_j leads to a new ARG (G_{j+1}). If G_{j+1} is *valid* and *compatible* with D, it is accepted with the MH probability in equation (2.9). Once the algorithm has reached convergence, samples from it are considered as samples from the CwR+.

Before discussing the transition types, let us first introduce some definitions:

Definition 4.3.1. *Valid proposal:* A transition type that does not remove an existing recombination event, i.e., none of the parent branches become NAM.

Definition 4.3.2. *Floating (pruned) lineage:* A branch that is detached from its parents.

Definition 4.3.3. *Active lineage:* A lineage between consecutive events that is not a floating lineage.

Definition 4.3.4. *Root:* A node that does not have a parent, i.e., all the ancestral sites on the node have already found their MRCA. An ARG, unlike a tree, might have multiple roots.

4.3.4 Transition 1. Subtree-Pruning-and-Regrafting operation

An SPR operation prunes a CA child lineage from G_j and reattaches the pruned lineage to the remaining ARG at an older time, resulting in G_{j+1}. To manage the reverse move, we do not allow the pruned lineage to experience recombination. As a result, to preserve reversibility, no existing recombination event should be canceled by an SPR move. If an SPR move would cancel an existing recombination event, the move is *invalid* and is rejected.

To employ the SPR, a CA event is randomly chosen. Then, one of its child nodes (d) is selected at random to prune from the ARG. In the following, the SPR steps are described.

4.3.4.1 Step 1: Detach d

Lineage d is detached from the ARG and is added to F, where F is an initially empty set of floating lineages. As a consequence of detaching d, a root node may cease to be a root. A node whose history has not been traced in G_j, may now carry some ancestral materials. Hence, the lineage is floating and requires reattaching; this node is also added to F. In contrast, a lineage may cease to carry any ancestral material, i.e., become a NAM lineage. NAM lineages are not allowed to evolve in our method and are discarded from the ARG.

4.3.4.2 Step 2: Reattach the floating lineages

Let t_d denote the initiation time of d. To reattach the floating lineages (including d) to the ARG, going up the ARG (backward in time) from t_d, for each time interval (t_i, t_{i+1}),

1. Set $t_i = t_d$.
2. Find the number of active lineages (k_i) and the number of floating lineages ($|F|$) at time interval (t_i, t_{i+1}).
3. Draw a time point (t') from an exponential distribution with rate

$$\lambda_i = \frac{|F|k_i + \binom{|F|}{2}}{2N}. \tag{4.8}$$

   - If $t_i + t' \geq t_{i+1}$, no new event is introduced in (t_i, t_{i+1}). Put $t_i = t_{i+1}$ and go to step 2.
   - If $t_i + t' < t_{i+1}$, a new CA event occurs at time $t_i + t'$. With probability

$$P_b = \frac{\binom{|F|}{2}/2N}{\lambda_i},$$

     this event occurs between two floating lineages, and with probability $1 - P_b$ it occurs between one of the active lineages and one of the floating lineages. Simulate the new event at $t_i + t'$. Update $t_i = t_i + t'$ and go to step 2.
4. Continue until $F = \phi$, and no lineage needs a further update.

In this process, if a recombination child or a recombination parent becomes NAM, the move is *invalid* and is rejected.

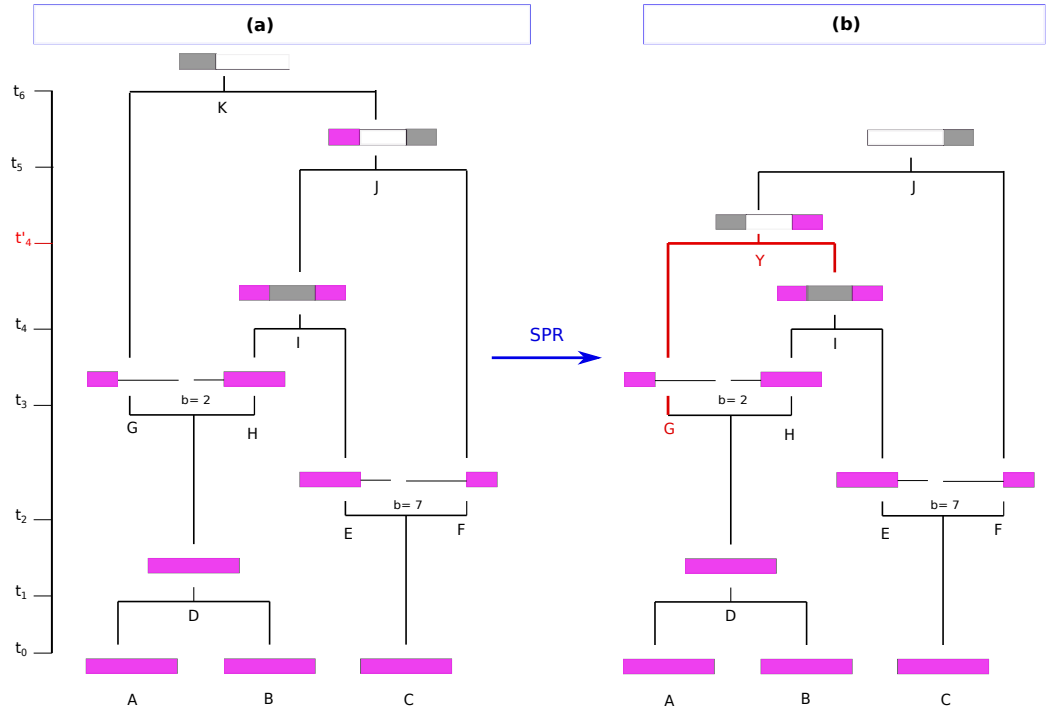


Figure 4.3: (a) The current ARG (G_j). (b) The proposed ARG after an SPR move: branch G is pruned from G_j and is rejoined to branch I at time t'_4.

4.3.4.3 Step 3: Transition probabilities

Forward transition probability

Here, we evaluate $Q(G_{j+1}|G_j)$. The number of new events in (t_i, t_{i+1}) is Poisson distributed with mean

$$\lambda_i = \frac{|F|k_i + \binom{|F|}{2}}{2N}. \tag{4.9}$$

Hence, with probability $P_0 = e^{-(t_{i+1}-t_i)\lambda_i}$ no new event occurs in (t_i, t_{i+1}) and at least one event occurs with probability $1 - P_0$. Moreover, the time (t_b) of a newly created event in (t_i, t_{i+1}) has a truncated exponential distribution with probability density function

$$f(t_b) = \frac{\lambda_i e^{-(t_b-t_i)\lambda_i}}{1 - e^{-(t_{i+1}-t_i)\lambda_i}}. \tag{4.10}$$

The product of the above terms for all the time intervals with $|F| > 0$ leads to the forward transition probability.

Reverse transition probability

The SPR rearrangement on G_j may create valid NAM lineages. These lineages do not carry any ancestral material and do not cancel any existing recombination event (e.g., branch J in Figure 4.3 (b)). These valid NAM lineages do not

affect the forward transition probabilities. However, they play an important role in the reverse transition probabilities. NAM lineages in the forward move act as floating lineages in the reverse move. Therefore, we need to identify these valid NAM lineages and find the probability of reattaching them moving from G_{j+1} to G_j. To evaluate $Q(G_j|G_{j+1})$, we apply the exact operations as explained in the forward transition calculation.

4.3.4.4 An SPR example

Figure 4.3 is an example of an SPR transition. Let $P(f)$ denote the forward transition probability, where $P(f) = 1$ initially. The SPR steps and the forward transition probability for this example are as follows:

- Choose a CA child with probability $1/8$. Here G is chosen, and is added to F, i.e., $F = \{G\}$. Also $P(f) = 1/8$.
- Rejoin G to the ARG at a time older than t_3 (the initiation time of G). For the time interval (t_3, t_4), $|F| = 1$, $k_3 = 3$. A time point (t') is generated from an exponential distribution with rate

  $$\lambda_3 = \frac{3}{2N}.$$

  Since $t' + t_3 > t_4$, no new event occurs at (t_3, t_4), and

  $$P(f) = P(f) \times e^{-(t_4 - t_3)\lambda_3}.$$

- For the next time interval (t_4, t_5), $|F| = 1$, $k_4 = 2$. A time point (t') is simulated from an exponential distribution with rate

  $$\lambda_4 = 1/N.$$

  $t'_4 = t' + t_4 < t_5$, therefore a new CA occurs in this interval. The CA is between G and a randomly chosen branch (here branch I) from the active lineages (I and F). The MRCA for segment $[1, 2]$ is reached and is located on the parent branch (Y). The forward transition probability is

  $$P(f) = P(f) \times \lambda_4 e^{-(t'_4 - t_4)\lambda_4}.$$

- For the time interval (t'_4, t_5), $|F| = 0$. Therefore, we only need to update the ancestral material of the affected branches. Here, the ancestral segment for branch J is updated. As seen from the Figure 4.3 (b), branch J becomes a root in G_{j+1}.

The reverse transition probability is

$$\begin{aligned} Q(G_{j+1}|G_j) = & \frac{1}{8} && \text{pick a lineage at random} \\ & \times e^{-(t_4-t_3)3/2N} && \text{no event in } (t_3, t_4) \\ & \times e^{-(t_5-t_4)/N} && \text{no event in } (t_4, t_5) \\ & \times \frac{e^{-(t_6-t_5)/2N}}{2N} && \text{unroot } J\text{, a CA between } G \text{ and } J \text{ at } t_6 = t_5 + t'. \end{aligned}$$

4.3.5 Transition 2. Remove an existing recombination

Uniformly at random, we select one of the N_r recombination events in G_j. We then pick one of the two parents (p_1 and p_2) of the recombination event. Assume p_1 is chosen, and c is the child branch of the recombination event. We look at the next event p_1 experiences, moving up the ARG. If it is a CA, p_1 is removed from the ARG, and c (the recombination child) follows the path given by the remaining parent, p_2. Otherwise, since the parent event of p_1 is a recombination event, removing p_1 would cancel that recombination event; therefore, the move is *invalid* and is rejected. Next, steps 2 and 3 of the SPR operation are applied to reattach the floating lineages (if any) and calculate the transition probabilities.

Figure 4.4 represents an example of recombination removal. The recombination event at time t_3 is removed. The child branch (D) follows the path of branch H. This transition changes the time of the GMRCA from t_6 to t_4. The forward transition probability is

$$Q(G_{j+1}|G_j) = \frac{1}{4},$$

because there are two recombination events in the ARG. The reverse transition is adding a new recombination event to the ARG. The reverse transition probability, therefore, is

$$\begin{aligned} Q(G_j|G_{j+1}) = & \frac{1}{7} && \text{pick a lineage } (D \text{ is chosen}) \\ & \times \frac{e^{-(t_3-t_1)}}{1-e^{-(t_4-t_1)}} && \text{choose a time } (t_3) \text{ for the new recombination event} \\ & \times \frac{1}{9} && \text{choose a recombination link on } D \\ & \times \frac{1}{2} && \text{choose one of the new parents to float (here } G) \\ & \times e^{-(t_4-t_3)3/2N} && \text{no new event in } (t_3, t_4),\ |F| = 1,\ k_3 = 3 \\ & \times e^{-(t_5-t_4)/N} && \text{no new event in } (t_4, t_5),\ |F| = 1,\ k_4 = 2 \\ & \times \frac{e^{-(t_6-t_5)/2N}}{2N} && \text{an event at } t = t_6,\ |F| = 2. \end{aligned}$$

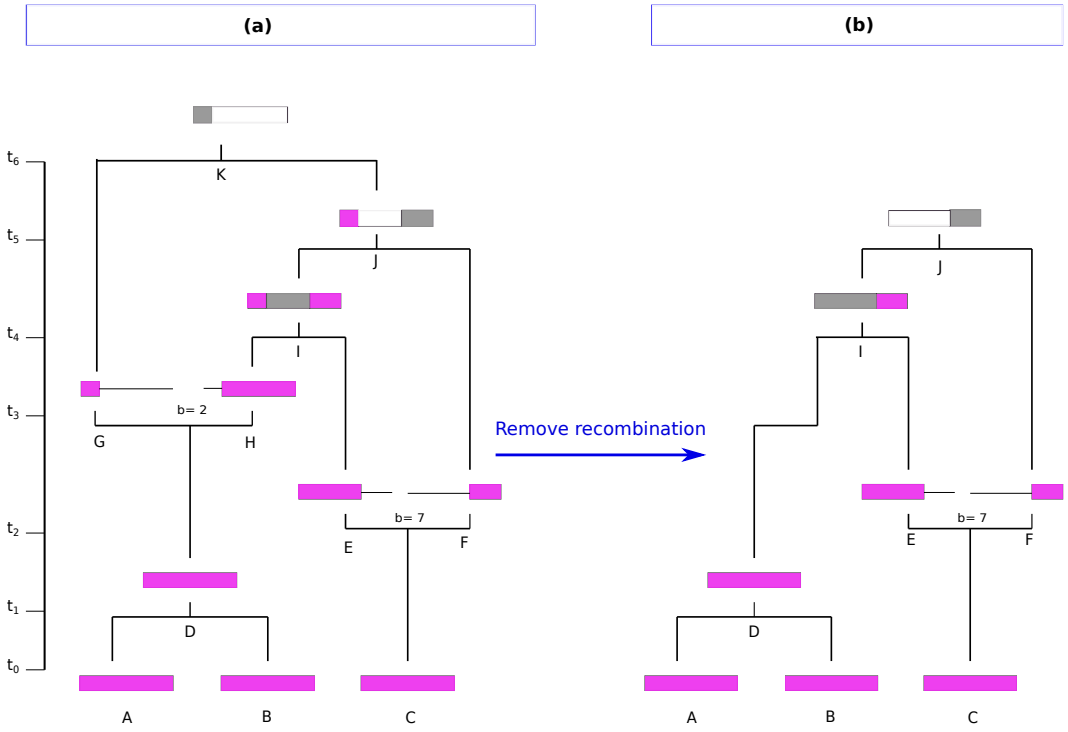


Figure 4.4: (a) The current ARG. (b) The ARG after removing the recombination event at time t_3.

4.3.6 Transition 3. Add a recombination event

We begin by choosing a branch (c) randomly from all the branches in G_j, excluding the roots. If t_c is the time of c, we simulate a time point t according to a truncated exponential distribution with rate 1 in the range (t_c, t_p), where t_p is the time of the parent of c. Next, a recombination breakpoint (b) is simulated uniformly at random on branch c. The branch c is split into two newly created branches (say u and v), one of which carries the ancestral material to the left of b, and the other contains the ancestral material to the right of b. One of the newly created branches (v) is selected randomly to follow the current path of c. The remaining branch, u, is a floating lineage and rejoins the ARG using the SPR steps 2 and 3.

Figure 4.5 is an example of this transition. First, branch I is randomly chosen, and a breakpoint ($b = 4$) is uniformly chosen among the recombination links. The time of the new recombination event is also generated from a truncated exponential distribution with rate 1 in the range (t_4, t_5). The branch is split into two lineages X and Y. Branch Y is randomly picked to follow path of branch I, and X becomes a floating lineage. Now we need to rejoin X to the ARG at somewhere older than t'_4. For the time interval (t'_4, t_5), $|F| = 1$, $k_4 = 3$. We generate a value (t') from an exponential with rate $3/2N$. Since $t' + t'_4 > t_5$, no new event occurs in this interval. As seen from the Figure, the branch I becomes a root. For the last time interval (t_5, ∞), both G and X are floating. These two branches merge at time t'_5. The resulting ARG in panel (b) is the proposed ARG, which has two roots.

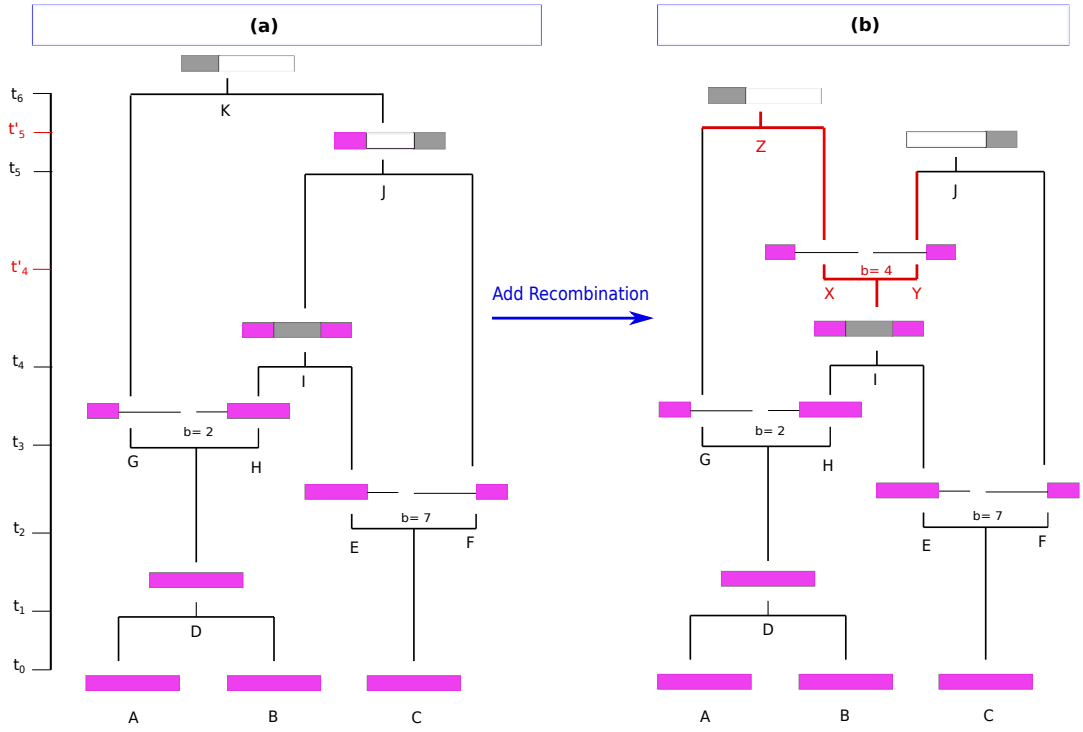


Figure 4.5: (a) The current ARG. (b) The proposed ARG (G_{j+1}) after adding a recombination event at time t_3.

The forward transition probability for this example is

$$
\begin{aligned}
Q(G_{j+1}|G_j) = & \frac{1}{10} && \text{pick a lineage } (I \text{ is chosen}) \\
& \times \frac{e^{-(t'_4 - t_4)}}{1 - e^{-(t_5 - t_4)}} && \text{choose a time } (t'_4) \text{ for the new recombination event} \\
& \times \frac{1}{9} && \text{choose a recombination link on } I \\
& \times \frac{1}{2} && \text{choose one of the new parents to float (here } X) \\
& \times e^{-(t_5 - t'_4)3/2N} && \text{no new event in } (t'_4, t_5),\ |F| = 1,\ k_4 = 3 \\
& \times \frac{e^{-(t'_5 - t_5)/2N}}{2N} && \text{an event at } t = t'_5,\ |F| = 2.
\end{aligned}
$$

The reverse transition removes a recombination event from G_{j+1}. For the example in Figure 4.5, branch X is randomly chosen and is removed from G_{j+1}. As a result, branch J is "unrooted" and becomes a floating lineage. Similarly, branch G is floating and can rejoin the ARG anywhere older than t_5. The reverse transition probability is

$$Q(G_j|G_{j+1}) = \frac{e^{-(t_6 - t_5)/2N}}{12N}.$$

4.3.7 Transition 4. Adjust event times

This transition updates all the node times according to the CwR. Going backward in time,

1. Set $t = 0$, k as the number of branches, and k' as the total number of recombination links.
2. Simulate a time point (t') from an exponential distribution with rate
$$\lambda = \frac{\binom{k}{2}}{2N} + rk'.$$
3. Simulate the next event time for G_j at $t + t'$.
4. Update $t = t + t'$, k, and k'. If $k > 1$, go to step 2. Otherwise, terminate the algorithm.

The proposed ARG G_{j+1} is accepted with probability

$$\frac{P(D|G_{j+1};\Theta)}{P(D|G_j;\Theta)}.$$

This ratio is a simplified version of the MH ratio in 2.9. It is simplified because this transition is based on the prior, and therefore the transition probabilities cancel with the prior ratio. In other words,

$$\frac{P(G_{j+1};\Theta)Q(G_j|G_{j+1})}{P(G_j;\Theta)Q(G_{j+1}|G_j)} = 1.$$

Hence, there is no need to calculate the transition probabilities.

4.3.8 Transition 5. Adjust recombination breakpoint

We randomly choose a recombination child (c). Then we choose a new breakpoint uniformly at random among the available links on c. Analogous to updating the ancestral lineages, the newly created floating lineages (if a root node ceases to be a root) are reattached to the ARG using the SPR operation. As an illustration, in Figure 4.6, the recombination breakpoint on branch I changes from 4 to 8. As a result, branches J and Z become floating and are rejoined to the ARG using the SPR.

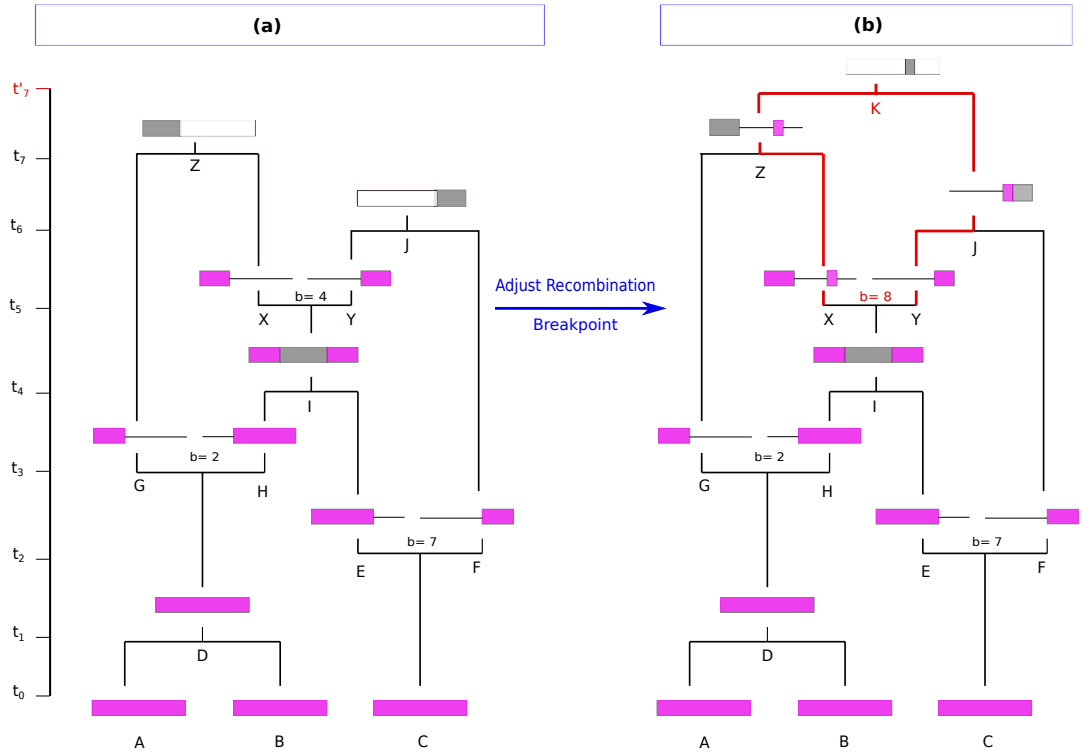


Figure 4.6: (a) The current ARG (G_j). (b) The proposed ARG (G_{j+1}) after adjusting the breakpoint of the recombination event at time t_5.

In general, the transition is symmetric if no NAM or floating lineages are created. Otherwise, the transition probabilities are calculated for the NAM and floating lineages. For instance, the transition probability for the example in Figure 4.6 is

$$\begin{aligned} Q(G_{j+1}|G_j) = & \frac{1}{3} && \text{pick a recombination event} \\ & \times \frac{1}{9} && \text{choose a recombination link on } I \\ & \times e^{-(t_7-t_6)/N} && \text{no new event in } (t_6, t_7),\ |F| = 1,\ k_6 = 2 \\ & \times \frac{e^{-(t'_7-t_7)/2N}}{2N} && \text{an event at } t = t'_7,\ |F| = 2 \end{aligned}$$

and $Q(G_j|G_{j+1}) = 1/27$.

4.3.9 Transition 6. Switch the order of two consecutive events.

Marjoram et al. (2000) introduced a proposal to rearrange the ARG locally. The idea is to switch the order of two consecutive events if they are not comparable. In other words, a pair of events is chosen, and if it is one of the four forms shown in Figure 4.7, the time of the events are swapped.

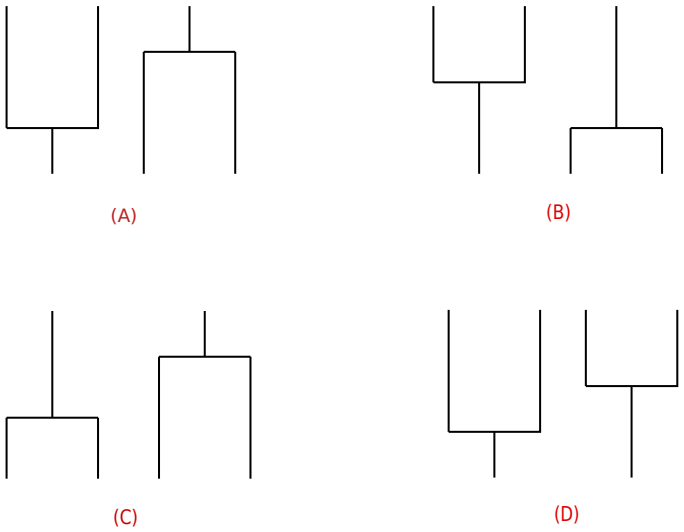


Figure 4.7: All the possible non-comparable consecutive pairs of events in an ARG.

This transition is symmetric because the number of events remains unchanged after the transition.

4.4 Results

We assessed our MCMC algorithm in inferring the CwR using simulated data. 50 replicate data sets are generated using *msprime*. Each data set consists of 5 DNA sequences comprising 6×10^4 sites. These data sets are generated under the CwR with $r = \mu = 10^{-8}$ and $N = 5 \times 10^3$. Then, we applied the MCMC algorithm to each data set for 2×10^5 sampling iterations. After discarding the first 10^5 outputs as burn-in, every 100th step is retained (1000 samples in total).

The algorithm is implemented in Python and ran using Spartan, a High Performance Computing System operated by Research Platform Services at The University of Melbourne (Lafayette et al., 2016), with 1 core and 10 GB RAM. The algorithm takes about 41 hours to run 2×10^5 sampling iterations for a single data set.

First, we assessed the convergence of the MCMC algorithm using trace plots, autocorrelations, and effective sample sizes (ESS) of different ARG parameters. We calculated the ESS by

$$\text{ESS} = \frac{T}{1 + \sum_{h=1}^{T'} \rho(h)},$$

where $\rho(h)$ is the autocorrelation at lag h, T the number of MCMC outputs, and T' the time when the autocorrelation first becomes negative (Kass et al., 1998). For a randomly chosen data set among the 50 data sets, Figure 4.8 shows the trace plots of the log-likelihood, log prior, log posterior, the total branch length, and the number of ancestral and non-ancestral recombination events

of the sampled ARGs. The x axis is the MCMC iterations after the burn-in and thinning. As seen from the Figure, none of the plots are stabilized, and the algorithm is very slow in exploring the state space. Figure 4.9 indicates that the autocorrelation reaches insignificant levels in approximately 50 iterations after thinning for the posterior, suggesting that the outputs are highly correlated. Also, there are 20 independent ARGs from the 1000 outputs, which is small. Therefore, convergence is not yet reached and requires longer runs.

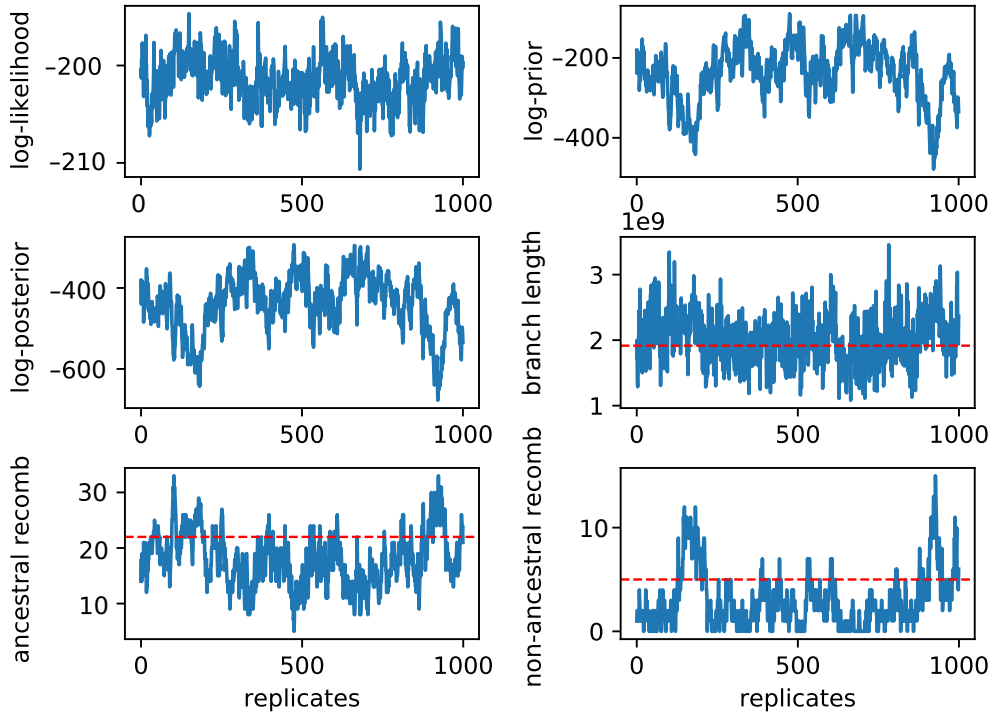


Figure 4.8: The trace plot of a randomly selected data set. The horizontal red line is the true value based on the output of *msprime*.

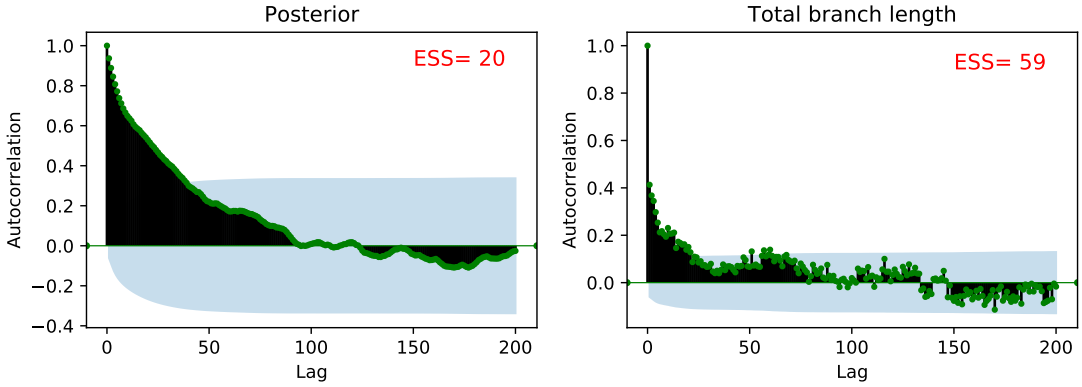


Figure 4.9: The autocorrelation in the posterior and the total branch length for the data set with trace plots in Figure 4.8. The blue shadow represents the 95% confidence interval for no correlation.

We investigated the ability of the MCMC method to approximate the CwR+ by comparing the inferred values of some ARG parameters against the truth for the 50 data sets. The top plots in Figure 4.10 show the true versus inferred values of the logarithm of the posterior probability and the total branch length of the ARG. We observe that the estimated values are approximately unbiased. However, most of the points lie below the line for the ancestral and non-ancestral recombination events (the bottom plots), indicating that the algorithm underestimates the number of recombination events. The underestimation is greater for the number of non-ancestral recombinations and worse for more recombinations.

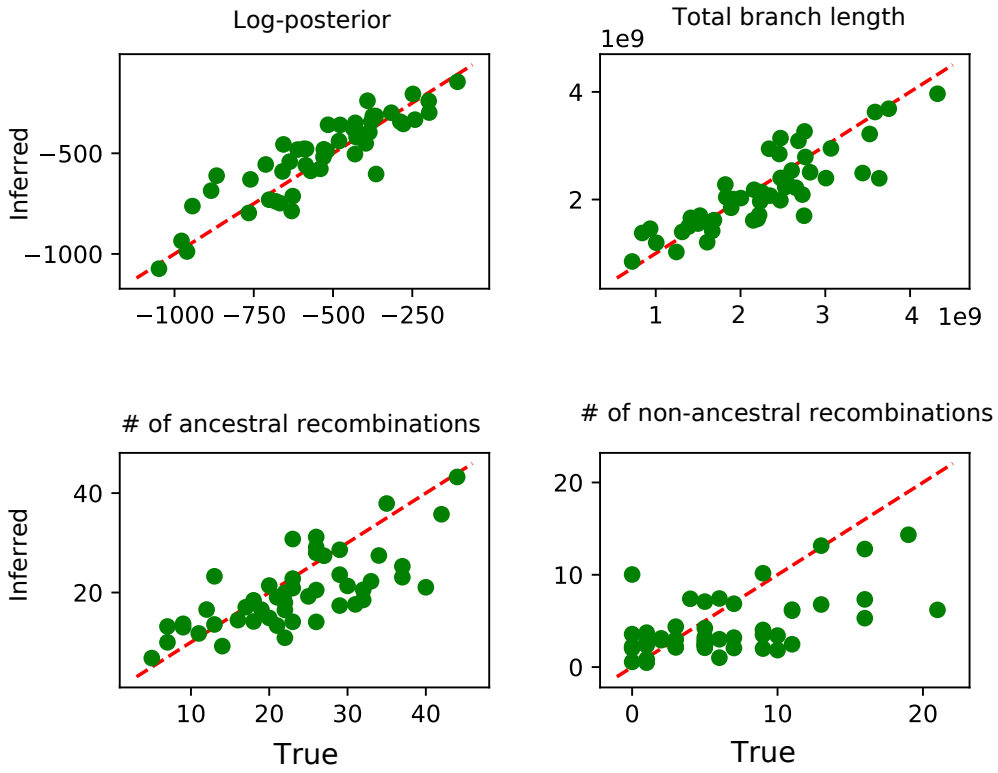


Figure 4.10: Estimation of the ARG features by the MCMC algorithm under *FTS*.

The acceptance rate is high (about 0.37). The individual acceptance rate for each proposal is shown in Figure 4.11. For each proposal, we categorized the rejections into MH, incompatible, and invalid categories. It is worth noting that, at each iteration, the six transition proposals are chosen with probability $0.1, 0.1, 0.1, 0.5, 0.1,$ and 0.1, respectively.

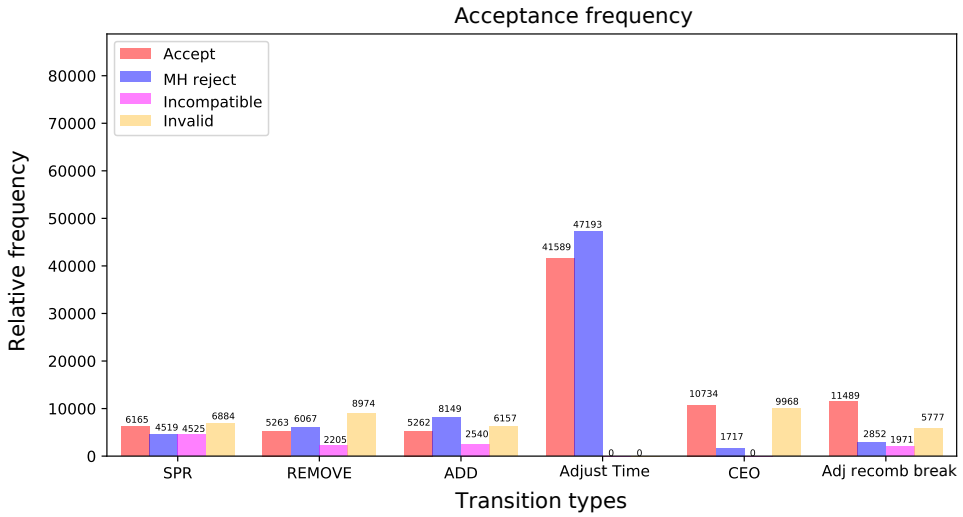


Figure 4.11: The acceptance frequencies of a selected data set. The x axis is the transition types: SPR, REMOVE (remove a recombination event), ADD (add a recombination event), adjust event times, CEO (switch the order of two consecutive events), and adjust recombination breakpoints.

In conclusion, unfortunately, the performance of the algorithm is unsatisfactory. It fails to estimate some of the ARG features accurately and is slow in convergence. In the next section, some potential reasons for this poor performance are discussed.

4.5 Limitations and complexities of *FTS*

4.5.1 Computationally-intensive updates

Any changes in the ancestral material of a lineage result in changes in its parent lineages and the tree branches. It is necessary to update the parents of all the branches that experience a change. As an illustration, removing the recombination event at time t_3 in Figure 4.4 causes significant changes in the records for lineages G, H, K, I, and J. As seen from Figure 4.4 *(b)*, the first three lineages are removed due to the recombination removal. Before the transition, lineage I carries ancestral material in segments $[1, 2]$ and $[8, 10]$, whereas it carries ancestral material in segment $[8, 10]$ after the transition. Similarly, the ancestral material changes on lineage J.

The efficiency of updating these changes on the ARG branches depends directly on the data structure. The *FTS* contains repeated information in different tables. These redundancies force the MCMC algorithm to spend a significant amount of time updating a single piece of information more than once. For instance, since a tree branch is a subset of an ARG branch, all the information in the former already exists in the latter. Therefore, whenever an ARG branch in the *ASC1* table is updated, we need to repeat the process for the corresponding tree branch in the *CR* table.

Also, in the likelihood evaluation algorithm, the child sequences are compared against each other site by site to identify mutations. The computational complexity of this costly operation could be reduced if *FTS* included the mutation information.

4.5.2 MRCA relocations

Another complexity we encounter in applying the above transitions is that some MRCAs of the proposed graph may be attained more recently or later than the current MRCA. For example, in Figure 4.4, the MRCA of segment $[1, 2]$ is at time t_6 before the transition, and is relocated to t_4 after the removal. We handle these alterations using *FTS* by counting the number of extant segments for a given interval of ancestral material and updating the *MRCA* table accordingly. However, this approach is expensive, given that many such fragments need to be monitored even for a local rearrangement on an ARG. Due to the structure of the *FTS*, it is not possible to apply an alternative strategy to reduce the cost of this operation.

4.5.3 Incompatible and invalid proposals

The MCMC algorithm delays the rejection of incompatible proposals to the last step; at each MCMC iteration,

1. a proposal move is chosen,
2. the transition move is applied to G_j. All the changes are addressed, resulting in G_{j+1},
3. the likelihood and the prior are evaluated. If G_{j+1} is not compatible with D, it is rejected.

This strategy wastes computation because many proposals (especially for large data sets) are incompatible. It would be more efficient to reject the incompatible moves immediately after they are proposed. However, since the *FTS* does not include the mutations, there is no way to identify such proposals earlier than the likelihood evaluation step.

A large portion of the proposed ARGs is rejected due to being *invalid* (see Figure 4.11 (yellow bars)). None of the existing proposals remove recombinations. As a result, most of the proposed arrangements of recombination hotspots would tend to be rejected, which prevents the algorithm from mixing well. A transition move that allows to cancel and add recombination events would help faster convergence.

4.6 Discussion

In this chapter, a new data structure, *FTS*, is introduced to represent the ARG. We then designed an MCMC algorithm using the *FTS* and evaluated the results using simulation data. Unfortunately, the MCMC algorithm does not perform

well, and even for a handful of sequences, the algorithm does not converge after 2×10^5 MCMC iterations. Running the MCMC with longer iterations is computationally expensive.

Some potential reasons for this poor performance have been discussed, including redundancies in the *FTS*, the complexities caused by the MRCA rearrangements, and the way the data structure deals with the incompatible proposals. Moreover, the transition proposals make small changes to the ARG so that exploring the vast ARG space is slow.

In the next chapter, I introduce a new data structure that overcomes some of these drawbacks.

Chapter 5

Improved data structure and MCMC algorithm

The poor performance of the MCMC algorithm using the *FTS* highlights the need for a new data structure and some new proposal transitions that allow for bigger changes on the ARG. In section 5.1, to remedy the shortcomings of the *FTS*, we introduce a new data structure that also exploits the efficiency of TS. In section 5.2, we define some notations to be used in the rest of the chapter. An algorithm is introduced in section 5.3 to build an ARG from the observed data. Section 5.4 discusses algorithms to calculate the likelihood and the prior. In section 5.5, some transition types are introduced that will be used to create a new MCMC algorithm.

5.1 Augmented TS

Augmented TS (ATS) is a collection of linked branches that connect the ancestral sequences. This connection facilitates the process of visiting each branch in the ARG. This property is particularly useful in MCMC because rearranging one branch may require visiting and rearranging many neighboring branches. Each branch consists of a sequence of segments that encode the genomic regions, which are also linked to extract and update the information efficiently. With this encoding, there is no need for storing marginal trees separately. We can recover any marginal tree from the *ATS*.

An important feature of the *ATS* is that it is an ARG with mutations. This data structure encodes the sequence data, which was not the case for *FTS*, and includes all the required information related to mutations. Section 5.1.4 discusses how this feature is helpful in the MCMC algorithm.

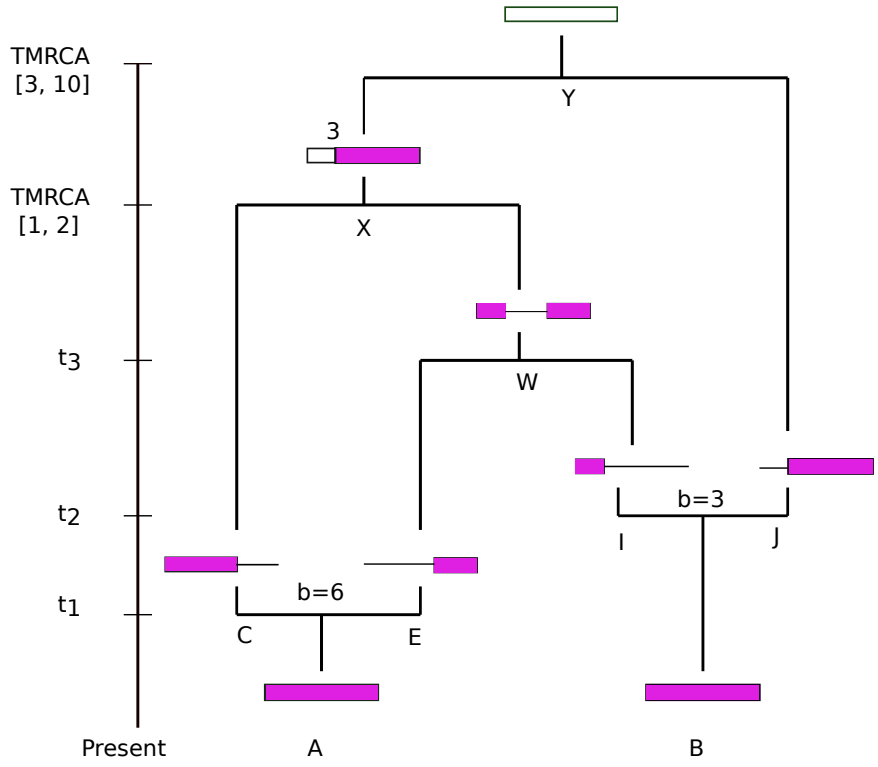


Figure 5.1: An ARG of two sequences with length ten sites. Branches C, E, I, and J are recombination parents, and branches W, X, and Y are the CA parents. The MRCA of segment $[1, 2]$ is reached at branch X and the MRCA of $[3, 10]$ at branch Y. The purple color is ancestral material.

5.1.1 Branch

Our definition of a branch in the *ATS* is a combination of "branch" and "node" in a commonly used terminology for a graph. An *ATS* branch contains:

- Time: the initiation time of the branch. For instance, in Figure 5.1, the time for branch W is t_3.
- Parents: if the event is a recombination, it has two parent branches. Otherwise, there is one parent (see Figure 5.1).
- Children: if the event is a CA, there are two children; otherwise, the child is the recombinant child.
- Breakpoint: if the event is a recombination, the recombination breakpoint is stored on the branch.
- Segments: a set of non-overlapping ancestral regions. See section 5.1.2.
- Mutations: the mutations on the branch. See section 5.1.4.

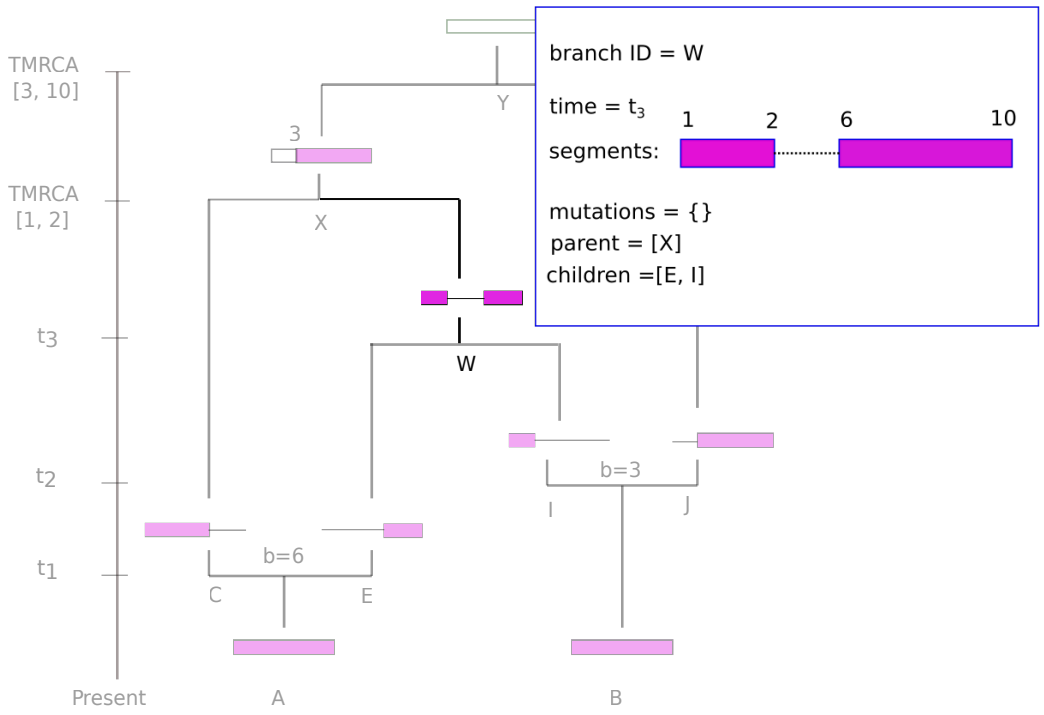


Figure 5.2: A branch in the *ATS*: Branch W is the parent of branches E and I.

Figure 5.2 illustrates a branch on the ARG. Branch W consists of a set of non-overlapping segments on intervals $[1, 2]$ and $[6, 10]$. The mutation set is empty, noting that there is no mutation along the branch. The parent of branch W is X, and the children are E and I.

Note that we do not label the nodes. If needed, a node can be called by the branch above if the event is a CA. In the case of a recombination event, a node can be referred by the event time (not needed in practice).

5.1.2 Segment

A set of non-overlapping segments on a branch is represented using a linked list. A *segment* is the smallest component of the *ATS* and records:

- A genomic region: an interval defined by two endpoints.
- Branch label: the branch on which the segment exists.
- Descendant samples: a set of the sequences that inherit the segment.

In Figure 5.3, z is a segment that records the ancestral material in the range $[6, 10]$. The segment x is the previous segment on the linked list, the descendent sample for z is sequence A, and the branch on which z exists is W.

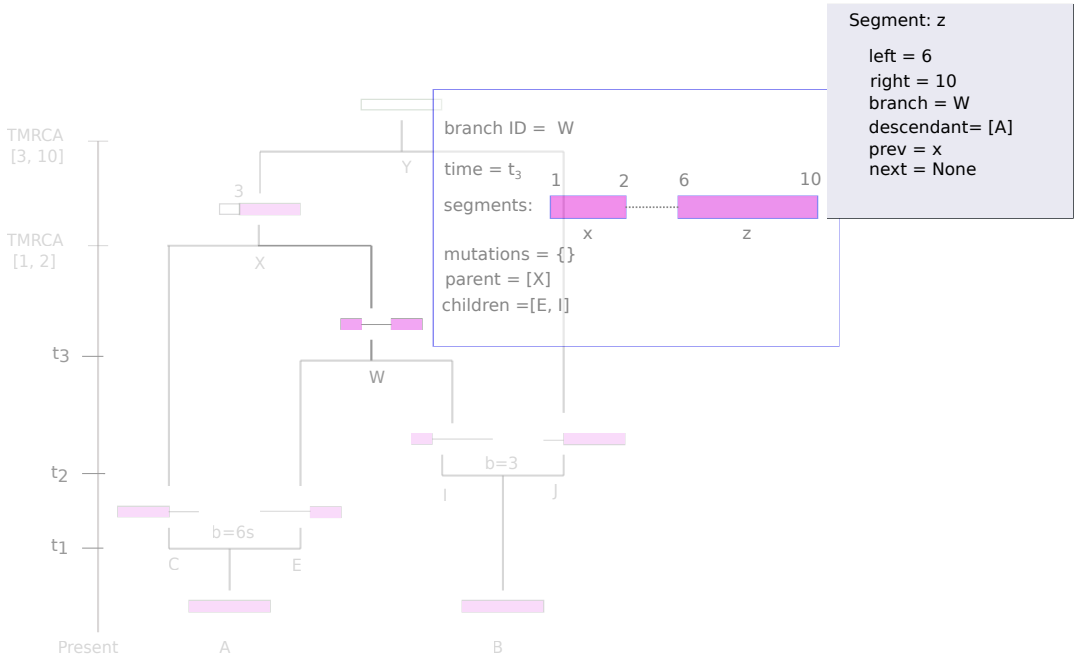


Figure 5.3: A segment: z is a segment of ancestral material in the range $[6, 10]$.

5.1.3 New format for representing DNA sequences

Before describing the *ATS* with mutations, we discuss our format for representing a set of DNA sequences. Commonly, D is represented in the form of a $n \times m$ matrix (Figure 5.4 *(a)*). Suppose the ancestral and derived alleles are known, specifically, 0 is ancestral and 1 is derived. An equivalent representation for D consists of a list of the sequences that carry the derived allele for each SNP. Let us denote the sequences that carry the derived allele for the SNP at site s as D_s. As seen from Figure 5.4 *(b)*, for the SNP at site 3 only A and B are recorded, which implies that these sequences carry the derived allele, and sequence C carries the ancestral allele.

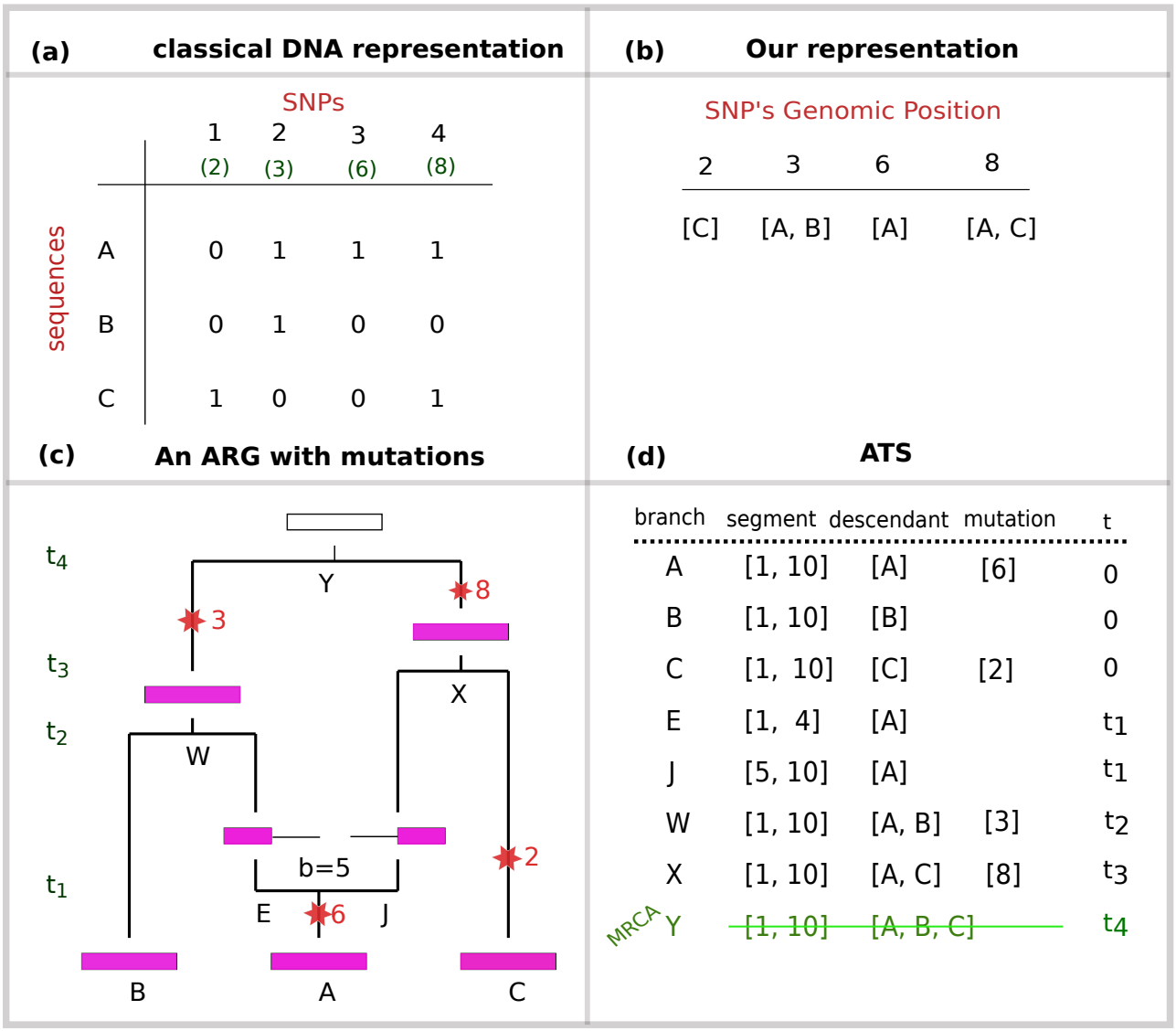


Figure 5.4: *(a)*: A classical representation of three sequences (A, B, and C) with 10 sites, four of which are segregating at sites 2, 3, 6, and 8. *(b)*: An equivalent representation of the DNA sequences in (a). For each SNP, the genomic position and the sequences that carry the derived allele are recorded. *(c)*: An ARG for D with mutations. The stars represent mutations for the SNPs at the locations indicated by red integers. *(d)*: The *ATS* representation of the ARG.

The new data format facilitates locating and identifying mutations on an ARG. For instance, from panel (b) in Figure 5.4, we can see that the sequences A and C carry the derived allele for SNP at site 8 (in short, SNP 8). Therefore, the most recent possible mutation position for this SNP is the first branch (branch X) that results from merging sequences A and C (going backward in time).

5.1.4 ATS is an ARG with mutation

Mutation information for each SNP is recorded in *ATS* branches. In general, for a SNP, the branch on which the mutation can be placed is not unique. However, in the *ATS*, a mutation is always located at the lowest possible branch. Thus, the branch that is the immediate ancestor of D_s records the mutation information for SNP s. For example, in Figure 5.4, the mutation on branch A (SNP 6) can also be located on branch J. However, branch A always (in all MCMC iterations) includes the mutation at site 6.

The *ATS*, as an ARG with mutations, assists the MCMC algorithm in several ways. Incompatible proposals can be identified immediately after they are introduced (in the update step of the MCMC algorithm). By comparing D_s and the samples descending from a segment, we can spot incompatible moves, see section 5.5.1.4 for details. Moreover, the likelihood evaluation can be done without comparing DNA sequences against each other. Instead, we use the mutation information recorded in the *ATS*. Lastly, we do not need to count extant sequences to locate new MRCAs. With information about the descendent samples for a segment, we can identify if the segment has reached its MRCA. Figure 5.4 (d) demonstrates the ARG in panel (a) in terms of *ATS* branches and segments. From the Figure, the MRCA for the genomic interval $[1, 10]$ takes place on branch Y, where all samples (A, B, and C) are merged.

5.2 Notation

Before we proceed to present the MCMC algorithm using the *ATS*, we require some notation.

Table 5.1: Table of notation for chapter 5, excluding the symbols in Table B.1.

Segment (y):	
l_y, r_y	left and right genomic end-sites
d_y	descendant samples
u_y	branch on which the segment exists
y_b, y_a	previous and next segments on u_y
Branch (u):	
p_{1u}, p_{2u}	left and right parents
c_{1u}, c_{2u}	left and right children
Y_u	the chain of segments on u
M_u	the set of mutations on u
b_u	recombination breakpoint
t_u	initiation time
S_u	the set of SNPs that the segments in Y_u cover, i.e., for a SNP at site s, if $\exists\ y \in Y_u$ that $s \in y$, then $s \in S_u$
The likelihood (section 5.4)	
D_s	the set of sequences that carry the derived allele for the SNP at site s
A_l	total branch length of an ARG
M	total number of mutations on an ARG
l_m	the branch length of all the branches on which mutation m could have fallen
U	the set of all the branches with at least one mutation
m_u	the set of mutations on branch u
S_v	genomic interval of branch v
Proposal distribution (section 5.5)	
F	the set of floating lineages
Z	the set of potential reattachment lineages
S_d	the set of SNPs on branch d that have not yet been mutated
$\lvert B_j \rvert$	the number of branches in G_j, excluding the roots
F'	the set of partially floating lineages
V	the set of branches required to visit
k'_e	the number of eligible links

Given the above notation, the following are some *ATS* properties:

- If u is a leaf, $c_{1u} = c_{2u} = \phi$.
- If u is a root, $p_{1u} = p_{2u} = \phi$.
- If u is a child of a CA event, $p_{1u} = p_{2u} = p_u$.
- If u is NAM, $u = \phi$.
- If u is a child of a recombination event, $p_{1u} \neq p_{2u}$.
- If u is a parent of a recombination event, $c_{1u} = c_{2u} = c_u$.
- If u is a parent of a CA event, $c_{1u} \neq c_{2u}$.

5.3 MCMC step 1: Construct an initial ARG

To construct an initial ARG for D in *ATS* format, we introduce a heuristic algorithm similar to the one in section 4.3.1. Recall from section 4.3.1 that the algorithm consists of three operations: *Mutation*, *Recombination*, and *Coalescent*. Here, the algorithm has two operations, *Recombination* and *Coalescent*. The *Recombination operation* is identical to that in section 4.3.1, whereas the *Coalescent operation* is different and is reviewed below.

- ***Coalescent operation:*** A branch (u) is randomly chosen and is compared with all other existing branches. Let H denote all the branches that can be merged with u; the branches that retain compatibility under the ISM after coalescing with u. If $H \neq \phi$, the branch ($v \in H$) with the highest overlapping genomic material with u is chosen to merge with u. Let S denote a set of SNPs that have not mutated yet. The newly recognized mutations (if any) are placed on the parent branch of u and v and are removed from S.

The algorithm steps are similar to section 4.3.1 without the *Mutation operation*. As an illustration, we explain the algorithm procedure to construct the ARG in Figure 5.4 *(c)* from the data in panel *(b)*:

- Set $k = 3$, $k' = 27$, $t = 0$, and $S = \{2, 3, 6, 8\}$.
- For any SNP s, if $|D_s| = 1$, put the mutation on the relevant branch. For $s = 6$, $|D_6| = 1$, put the mutation 6 on branch A, and remove 6 from S. Similarly, put the mutation of SNP 2 on branch C. Now $S = \{3, 8\}$.
- A time point ($t' = t_1$) is generated from an exponential distribution with rate
$$\lambda_0 = \frac{\binom{k}{2}}{2N} + k'\rho = \frac{3}{2N} + 27r.$$
The event at t_1 is a recombination (probability $27r/\lambda_0$). Branch A and breakpoint 5 on A are randomly chosen. Split A into two branches E and J. Update $k = 4$, $k' = 26$ and $t = t_1$.
- Generate a number (t') from an exponential distribution with rate
$$\lambda_1 = \frac{3}{N} + 26r.$$
The next event at $t_2 = t + t'$ is a CA (probability $3/N\lambda_1$). Branch E is randomly chosen. The potential branches to coalesce with E are $F = \{J, B\}$. Note that branch C cannot be chosen because of SNP 3. Branch B has the highest overlapping genomic material with E, and is chosen to coalesce with E. The parent branch is W with descendant sample (A, B). For SNP 3, $D_3 = (A, B)$, therefore, the mutation occurs on branch W. Update $k = 3$, $k' = 23$, $S = \{8\}$, and $t = t_2$.

- The next time point is generated from an exponential distribution with rate
$$\lambda_2 = \frac{3}{2N} + 23r.$$
The next event at time $t_3 = t + t'$ is a CA (probability $3/(2N\lambda_2)$). Branch C is randomly chosen, and is coalesced with branch J. The descendant sample for the parent (X) is (A, C). For SNP 8, $D_8 = (A, C)$, therefore the mutation is recorded on branch X. Update $k = 2$, $k' = 18$, $S = \phi$, and $t = t_3$.

- Lastly, a number is simulated from an exponential distribution with rate
$$\lambda_3 = \frac{1}{2N} + 18r.$$
The event at $t_4 = t + t'$ is a CA (probability $1/2N\lambda_3$). The branches W and X are merged. For the parent branch (Y), the descendant samples are (A, B, C), hence Y is MRCA for the segment $[1, 10]$, so the segment is deleted from branch Y.

5.4 MCMC step 2: Likelihood and prior evaluation

5.4.1 The likelihood

Assume that an *ATS* is given for D. We evaluate the likelihood by

1. calculating the total branch length of the ARG, and
2. computing the potential branch length for branches with at least one mutation, i.e., the length of all the branches on which a mutation could have taken place.

Assume A_l and M are the total branch length (in generations) of the given ARG and the total number of mutations, respectively. The number of mutations has a Poisson distribution with mean $A_l\mu$. For evaluating the likelihood, first, the probability of exactly M mutations on the ARG is calculated by

$$P(M) = \frac{e^{-A_l\mu}(A_l\mu)^M}{M!}. \tag{5.1}$$

Then, the probability of observing the exact mutation pattern is calculated. If m is a mutation taking place on branch u, the branch length l_m can be found from all the branches on which mutation m could have fallen. The probability of m taking place on branch u is l_m/A_l. Consequently, for all the mutations,

$$\prod_{u \in U} \prod_{m \in m_u} \frac{l_m}{A_l}, \tag{5.2}$$

where U is a set of branches at which at least one mutation taken place, and m_u is a set of mutations on branch u. Multiplying (5.1) and (5.2) results in the

likelihood,

$$\begin{aligned} P(D|G,\Theta) &= \frac{e^{-A_l\mu}(A_l\mu)^M}{M!}\prod_{u\in U}\prod_{m\in m_u}\frac{l_m\mu}{A_l\mu} \\ &= \prod_{v\in T}\frac{e^{-l_vS_v\mu}(l_vS_v\mu)^{|m_v|}}{M!}\times\left(\frac{l_v\mu}{l_vS_v\mu}\right)^{|m_v|} \\ &= \prod_{v\in T}\frac{e^{-l_vS_v\mu}(l_v\mu)^{|m_v|}}{M!}, \end{aligned} \tag{5.3}$$

where T is a set of all tree branches in the form of TS (i.e., identical subtrees are recorded only once). Note that $A_l = \sum l_vS_v$ where l_v and S_v are the branch length and genomic interval of branch v.

Since the denominator in (5.3) does not depend on μ nor the ARG, it will cancel out in the MH ratio in the MCMC algorithm. Therefore, we evaluate the likelihood with an unnormalized version of (5.3), that is

$$e^{-A_l\mu}(A_l\mu)^M\prod_{u\in U}\prod_{m\in m_u}\frac{l_m}{A_l}. \tag{5.4}$$

Let us calculate the likelihood for the ARG in Figure 5.4 *(c)*. We first calculate the total branch length of the ARG,

$$A_l = 10t_2 + 10t_1 + 4(t_2 - t_1) + 6(t_3 - t_1) + 10t_3 + 10(t_4 - t_2) + 10(t_4 - t_3).$$

The likelihood is

$$P(D|G;\Theta) = e^{-A_l\mu}(A_l\mu)^M \times \frac{{t_3}^2(t_4 - t_2)(t_4 - t_3)}{{A_l}^4}.$$

5.4.2 The prior

The information needed to compute the prior is also encoded in the *ATS*. The prior evaluation is the same as in section 4.3.2.2.

5.5 MCMC step 3: Explore ARG space

In this section, we introduce some proposals to update the ARG:

- SPR operation.
- Removing an existing recombination event.
- Adding a recombination.
- Modifying the node times.
- Adjusting a recombination breakpoint.
- Resampling a sub-graph of ARG.

Note that the SPR, remove recombination, add recombination, and adjust recombination breakpoint differ from those introduced in section 4.3.3.

5.5.1 Transition 1. Subtree-Pruning-and-Regrafting

A CA event with parent p and children d and c is randomly chosen. Then, randomly a child among the children, say d, is selected to detach from the ARG. The algorithm steps are outlined below.

5.5.1.1 Step 1: Detach d

Branch d is detached from the ARG. If p is not a root (Figure 5.5 (a)), connect c to the parent of p and take d as a floating lineage. Otherwise, both d and c are floating lineages (Figure 5.5 (b)). Add the floating lineage(s) to F, which is an initially empty set of floating lineages.

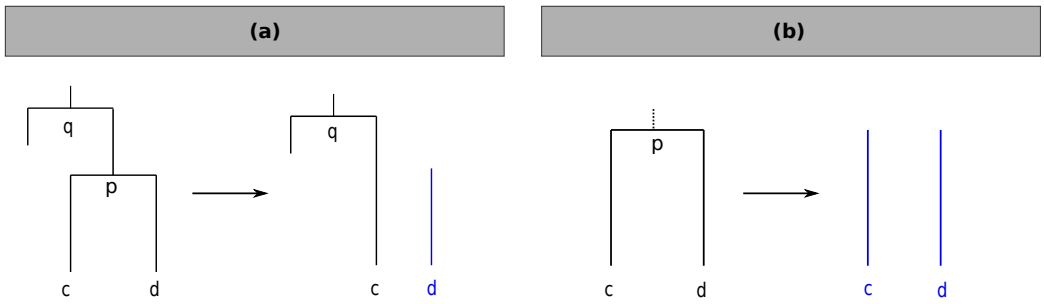


Figure 5.5: Detach branch d. *(a)*: The parent p is not a root. *(b)*: p is a root. Blue branches are floating lineages.

The forward transition probability for this step is $1/2N_c$, where N_c is the number of all the CA events on G_j.

5.5.1.2 Step 2: Update the ancestral material

We update the ancestral material of all the branches on the path of d to the GMRCA. As we move up the ARG to update the ancestral material, a root may cease to be a root. These cases are treated as floating lineages and are added to F (Figure 5.6 *(b)*). On the other hand, as we update the ancestral material of the branches going backward in time, a branch may become a NAM lineage. Eventually, such branches are discarded from the ARG. At this stage, however, since it is not clear that the proposed ARG (G_{j+1}) will be accepted, these *NAM* lineages are retained. Figures 5.6 and 5.7 show the process of applying the SPR on an ARG.

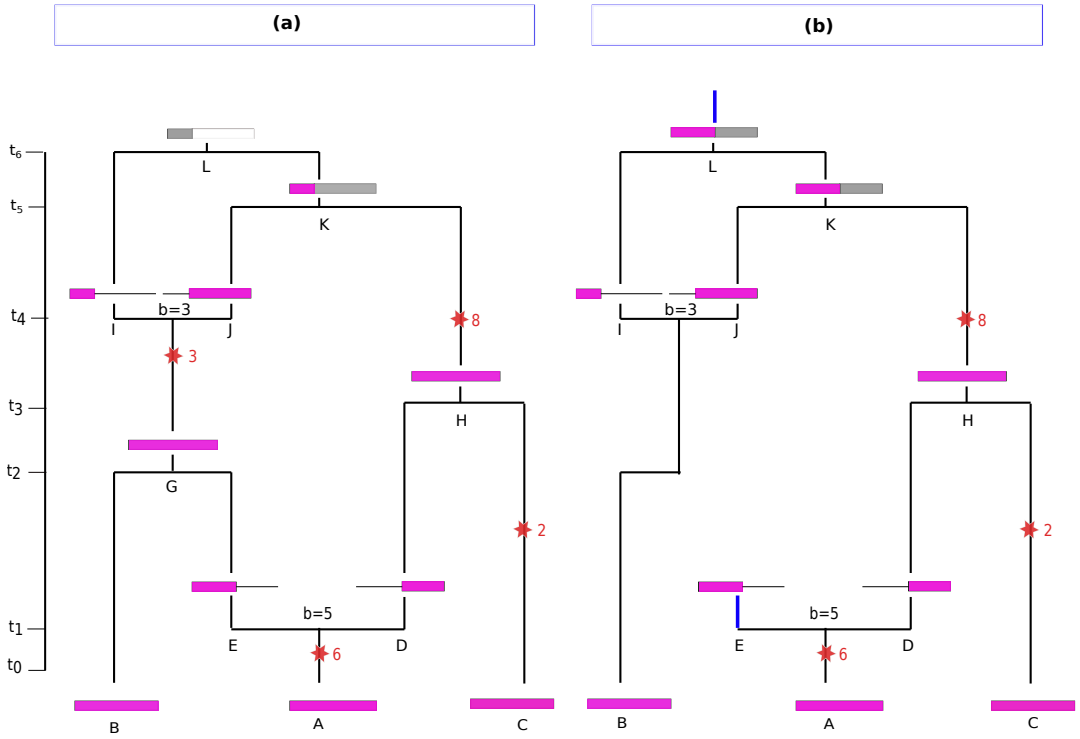


Figure 5.6: SPR steps 1 and 2. *(a)*: An ARG (G_j) compatible with D in Figure 5.4 *(b)*. *(b)*: Branch E is randomly chosen with probability $1/8$, and is pruned from G_j. Since the parent of E (branch G) was not a root, only E is added to the floating set ($F = \{E\}$). Next, the ancestral material on branches I, J, K, and L are updated. As seen, branch L ceases to be a root, so it is added to F; $F = \{E, L\}$.

5.5.1.3 Step 3: Reattach the floating lineages

All the floating lineages are reattached to the ARG in the following fashion:

1. Choose the branch with the lowest time (say u) in F. It is essential to follow the time order. Otherwise, the transition is not reversible.
2. Choose a branch (v) randomly among those to which u can reattach (Z), which requires that
   - $t_v > t_u$ and $v \neq \phi$ (i.e., is not a newly created NAM), or
   - $t_v < t_u$ and $t_{p_v} > t_u$, where t_{p_v} is the time of the parent of v.
3. If v is a root, the time t at which u and v coalesce is simulated from an exponential distribution with rate $1/2N$. Otherwise, the coalescence time is uniform in the interval $(\max(t_v, t_u), t_{p_v})$.
4. Coalesce u and v at t. Go to step 5.5.1.2.
5. If $F \neq \phi$, go to step 1. Otherwise, stop.

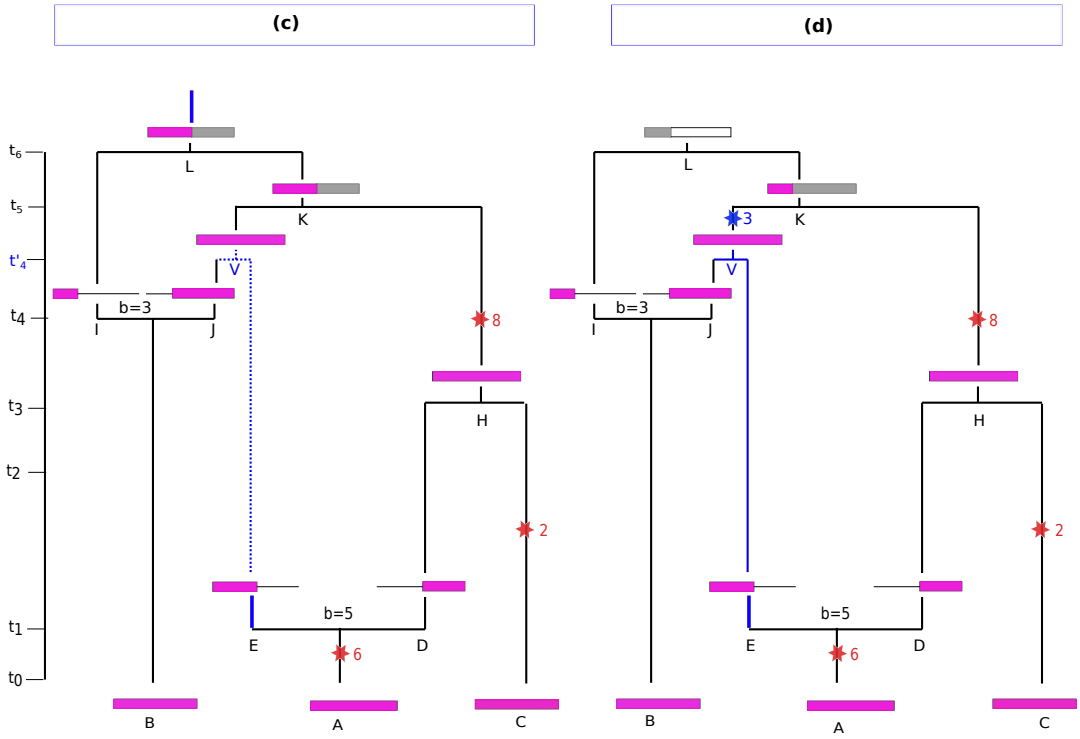


Figure 5.7: SPR steps 3 and 4. *(c)*: The branch with the lowest time (here, E) in F is chosen. A branch (J) among $Z = \{B, C, D, I, J, H, K, L\}$ is randomly picked to coalesce with E. The time of the new CA event (t'_4) is simulated from a uniform distribution with range (t_4, t_5). Update $F = \{L\}$. Note that if any branch other than $\{B, J\}$ is chosen, the move is incompatible. *(d)*: Update the ancestral material for branches V, K, and L. Branch L becomes a root and so is removed from F. Since $F = \phi$, the SPR step 3 is terminated. Now, we apply SPR step 4. The proposed ARG is valid because no recombination is canceled. To place the relevant mutations, note that $S_E = \{2, 3\}$ because branch E carries the segment $[1, 4]$. As seen from Figure 5.4 *(b)*, sites 2 and 3 within this segment are segregating. The SNP at $s = 2$ is already mutated on branch C, so $S_E = \{3\}$. Branch V contains segment y with $l_y = 3, r_y = 5, d_y = (A, B)$, and $s = 3 \in y$. From Figure 5.4, we observe that $D_3 = (A, B)$. $|w| = |d_y - D_3| = 0$, and $|D_3| = |d_y|$, hence the mutation at site 3 occurs on branch V. $S_E = \phi$, and so there are no other SNPs to be checked.

The forward transition probability for this step is

$$\prod_{u \in F} \frac{1}{|Z|} \left(I_v \frac{e^{-(t-t_l)/2N}}{2N} + (1 - I_v) \frac{1}{t_{p_v} - t_l} \right), \tag{5.5}$$

where t is the time of the new event, $t_l = \max\{t_u, t_v\}$, and I_v is 1 if v is a root, and 0 otherwise. The forward transition probability for the example of Figures 5.6 and 5.7 is

$$
\begin{aligned}
Q(G_{j+1}|G_j) = \frac{1}{8} & \quad \text{pick a lineage at random } (E \text{ is chosen}) \\
\times \frac{1}{8} & \quad \text{pick a lineage to rejoin } E \text{ to at random } (J \text{ is chosen}) \\
\times \frac{1}{t_5 - t_4} & \quad \text{generate a new time } (t_4') \text{ in } (t_4, t_5) \text{ at random}
\end{aligned}
$$

5.5.1.4 Step 4: Update mutations and check validity

Now that all the floating lineages have been reattached to the ARG, the validity and compatibility of G_{j+1} are evaluated.

Validity Check (VC)

G_{j+1} may contain some *NAM* lineages. If a child (or a parent) of a recombination event is *NAM*, the transition is not valid and is rejected.

Compatibility Check (CC)

Analogous to VC, we check whether G_{j+1} is compatible with D under the ISM. Let S_d denote the SNPs on branch d that have not been mutated earlier in the ARG (moving backward in time). Elements of S_d will be examined further up in the ARG to find their mutation branch. At any stage, if the proposed mutation is not compatible with the data, the transition is rejected.

For each SNP $s \in S_d$, as we move up the ARG, suppose a branch u, with segments Y_u contains s; that is, for a $y \in Y_u$, $s \in y$. Recall that d_y is the set of samples descending from segment y and D_s denotes a set of the sequences that carry the derived allele at site s. The move is incompatible at site s, if and only if $|d_y - D_s| > 0$ and $|D_s - d_y| > 0$. Moreover, if $|d_y - D_s| = 0$, and $|D_s| = |d_y|$, then u is the mutation branch for site s and s is added to M_u. The above operation is applied to all the sites in S_d (see Figure 5.7 *(d)*).

New roots and the reverse transition probability

If a *NAM* branch is the child of a CA event in G_j, it might be a new root in G_{j+1}. If it is a new root, the reverse transition probability needs to be calculated; we treat these new roots as floating lineages, but in the reverse move.

Assume u is a child branch of a CA event with parent p_u and sibling v in G_{j+1}. If u is *NAM* and exists in G_j, then u is a new root in G_{j+1}. For such branches, the reverse move is to rejoin u to v at time t_{p_u}. First, a branch is randomly selected among all the potential reattachment branches for u and then a time is chosen for the new event. Conditioning on whether or not p_u is a root, the new time is generated from an exponential or uniform distribution,

respectively. The reverse transition probability, therefore, is

$$\frac{1}{|Z|}\left[I_v \frac{e^{-(t_{p_u}-t_l)/2N}}{2N} + (1-I_v)\frac{1}{t_a - t_l}\right],$$

where $|Z|$ is the number of potential reattachment branches for u, t_a is the time of the parent of p_u, and $t_l = \max\{t_u, t_v\}$. The reverse transition probability for the example in Figures 5.6 and 5.7 is

$$Q(G_j|G_{j+1}) = \frac{1}{64(t_4 - t_1).}$$

5.5.1.5 Step 5: MH ratio

If G_{j+1} is compatible and valid, it is accepted with probability (2.9). If G_{j+1} is accepted, we remove all the NAM lineages in G_{j+1}. Otherwise, we discard G_{j+1} and retain G_j.

5.5.2 Transition 2. Remove a recombination event

Similarly to section 4.3.5, a recombination event is randomly chosen. Recall that p_1 and p_2 are the parent branches of the selected recombination event, and c is the recombination child. The branch p_1 is randomly picked among the parents, if the next event p_1 experiences (going backward in time) is a CA, p_1 is removed. Otherwise, the move is rejected. If p_1 is removed, c continues the path of p_2. The affected branches are modified by using SPR step 2 (section 5.5.1.2) to update the ancestral material of all the ancestors of c. Afterward, the SPR step 3 (section 5.5.1.3) is applied to reattach all the floating branches (if any) and calculate the relevant transition probabilities. To check the validity and compatibility of G_{j+1} as well as calculating the reverse probabilities, we apply SPR step 4 (section 5.5.1.4). If the move is accepted after the MH ratio, we remove the *NAM* lineages. Otherwise, G_j is retained.

Figure 5.8 illustrates a recombination removal. The forward and reverse transition probabilities for this move are

$$Q(G_{j+1}|G_j) = \frac{1}{4}, \tag{5.6}$$

and

$$
\begin{aligned}
Q(G_j|G_{j+1}) = & \frac{1}{7} && \text{pick a lineage } (G \text{ is chosen}) \\
& \times \frac{e^{-(t_4-t_2)/2N}}{2N(1-e^{-t_5-t_2/2N}} && \text{pick a time } (t_4) \text{ for the new recombination event} \\
& \times \frac{1}{9} && \text{pick a recombination link on } G \\
& \times \frac{1}{2} && \text{pick one of the new parents to float (here } I) \\
& \times \frac{1}{3} && \text{pick a lineage to rejoin } I \text{ to at random } (K \text{ is chosen}) \\
& \times \frac{e^{-(t_6-t_5)/2N}}{2N} && \text{pick a time } (t_6) \text{ for the new CA event.}
\end{aligned}
\tag{5.7}
$$

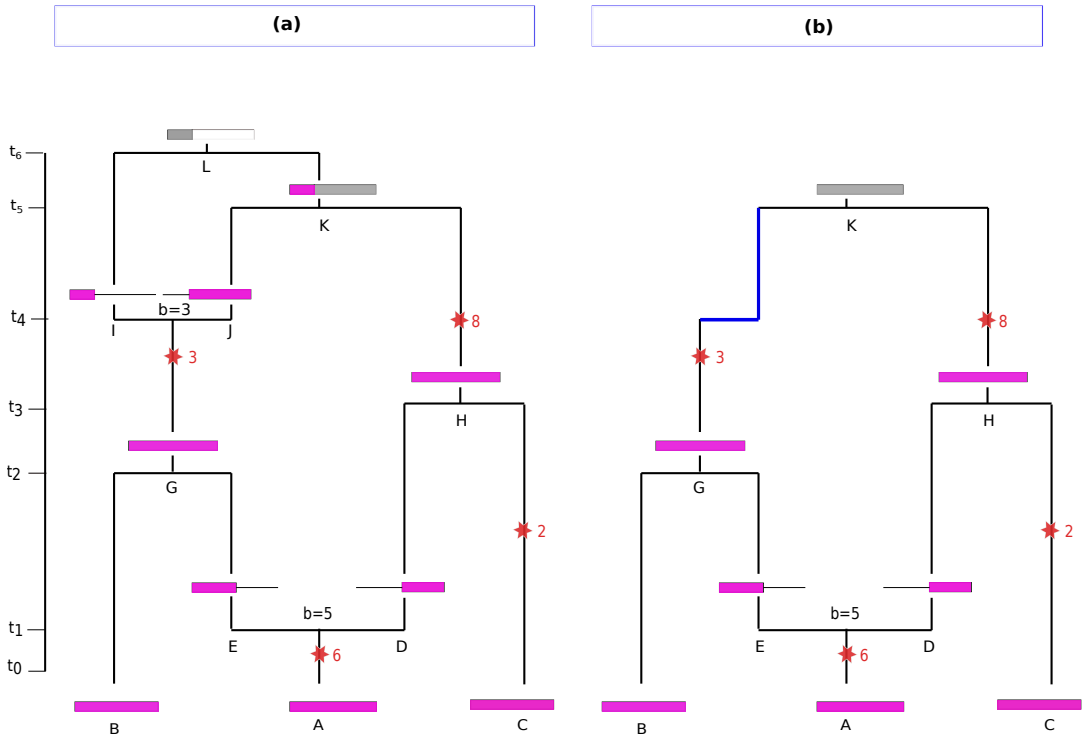


Figure 5.8: Recombination removal. *(a)* An ARG (G_j) compatible with D in Figure 5.4. *(b)* A recombination parent (I) is randomly chosen among branches I, J, E,and D. The parent I is removed, and G continues the path of J.

5.5.3 Transition 3. Add a recombination event

As stated in section 4.3.6, a branch (c) is randomly chosen, and a recombination is placed on it. One of the recombination parents follows the path given by c and the other is floating, and rejoins the ARG using the SPR operation. For the

above operations, the forward transition probability is

$$\frac{e^{-(t-t_c)/2N}}{4N|B_j|c_{k'}(1-e^{-(t_{p_c}-t_c)/2N})},$$

where $|B_j|$ is the number of branches, excluding the roots, in G_j, and $c_{k'}$ is the number of recombination links on branch c. The transition probability for rejoining the floating lineage(s) are given in (5.5).

To illustrate, assume the ARG in Figure 5.8 *(b)* is the current ARG. A recombination event is added to the ARG at time t_4, resulting in the proposed ARG in Figure 5.8 *(a)*. The forward and reverse transition probabilities are given in (5.7) and (5.6), respectively.

5.5.4 Transition 4. Adjust event times

The transition is identical to that in section 4.3.7.

5.5.5 Transition 5. Adjust a recombination breakpoint

Similar to section 4.3.8, a recombination child is selected at random, and the recombination breakpoint is updated. The rest of the algorithm is similar to the SPR steps from 5.5.1.2 to 5.5.1.5.

5.5.6 Transition 6. Resample a part of ARG according to the CwR

Kuhner et al. (2000) introduced a proposal distribution to rearrange a subtree of the ARG in proportion to the prior probability. More specifically, a branch is randomly selected to be pruned from the ARG and then reattached to the ARG according to the CwR. Hence, the pruned branch can undergo recombination, which is helpful for better MCMC mixing. We call this transition move the "Kuhner move." Before providing a full description of the rearrangement scheme, some definitions are needed.

Definition 5.5.1. *Eligible links:* The newly created recombination links on a branch in G_{j+1}, which were non-ancestral in G_j.

Definition 5.5.2. *Partially floating lineage:* A lineage that is not floating but contains some *eligible links*. Note that a partially floating lineage is an active lineage.

We use F and F' to denote a set of floating and partially floating lineages. Also, V is a set of branches that require checking. Lastly, let k'_e denote the number of eligible links.

5.5.6.1 Step 1: Detach d

The transition starts by randomly selecting a branch (d) among the branches in G_j, excluding the roots. Then d is detached from the ARG and is added to

F. All the affected branches, including c, p_d, and d, are added to V, so that it contains all the branches that might be changed. In addition, we add the number of recombination links on d to k'_e. If p_d is a root, c will be floating if $t_c < t_d$.

5.5.6.2 Step 2: Reattach the floating lineages

From t_d root-ward, for each time interval, the possibility of a new event is checked based on the CwR. If a new event occurs, it is either a recombination or a CA event. A new recombination event can happen on a floating or partially floating lineage. The CA event may merge two floating lineages or one floating and one active lineage. To reattach all the floating lineages, for all the time intervals with $F \neq \phi$ or $F' \neq \phi$, we apply the following procedure.

1. Set $t_i = t_d$.
2. Find the number of floating lineages ($|F|$), the number of partially floating lineages ($|F'|$), and the number of active lineages (k_i) immediately after t_i (root-ward).
3. Simulate a time point (t') based on an exponential distribution with rate

$$\lambda_i = \frac{|F|k_i + \binom{|F|}{2}}{2N} + rk'_e. \tag{5.8}$$

4. If $t_i + t' < t_{i+1}$, a new event occurs in the time interval (t_i, t_{i+1}). The event is a CA with probability

$$P_c = \frac{\binom{|F|}{2} + |F|k_i}{\lambda_i},$$

or recombination with probability

$$P_r = \frac{rk'_e}{\lambda_i}.$$

   (a) If it is a CA, with probability

$$P_f = \frac{\binom{|F|}{2}}{P_c},$$

   this event occurs between two floating lineages. Otherwise, with probability $1 - P_f$ the event occurs between one floating and one active lineage. Choose the lineages at random and place the new CA event at time $t_i + t'$. Update the relevant mutations and check the compatibility. If the move is incompatible with D, terminate the algorithm. Otherwise, update $t_i = t_i + t'$, add the affected branches to V, update k'_e, and go to step 2.

   (b) If the event is a recombination, one lineage (v) is chosen among all the lineages in F and F' proportional to their number of eligible

links. If $v \in F$, split v from a random genomic position and add both newly built branches to F. If $v \in F'$, split the branch from a genomic position randomly chosen among the eligible links. One of the resulting parents is randomly chosen and is added to F, and the other new branch follows the path of v. This event occurs at $t_i + t'$. Add the affected branches to V, update $t_i = t_i + t'$ and k'_e, and go to step 2.

5. If $t_i + t' \geq t_{i+1}$, no new event is introduced in (t_i, t_{i+1}). Update the relevant mutations and check compatibility using the *CC* operation in section 5.5.1.4. If the proposal is not compatible with D, terminate the algorithm. Otherwise, put $t_i = t_{i+1}$, add the affected branches to V, update k'_e, and go to step 2.

6. Continue until $V = \phi$.

5.5.6.3 Step 3: Transition probabilities

An advantage of re-simulating under the prior is the ease of transition probability calculation. Assume $|B_j|$ and $|B_{j+1}|$ are the number of branches (excluding the roots) on G_j and G_{j+1}, respectively. The reverse to forward transition probability ratio for choosing a branch to detach is $|B_j|/|B_{j+1}|$. This is the only step that is not driven by the CwR. The rest of the transition probability ratio cancel with the prior (see Appendix A), that is

$$\frac{Q(G_j|G_{j+1})P(G_{j+1})}{Q(G_{j+1}|G_j)P(G_j)} = 1. \tag{5.9}$$

Accordingly, the MH ratio for the Kuhner move reduces to

$$A = \min\{1, \frac{P(D|G_{j+1})|B_j|}{P(D|G_j)|B_{j+1}|}\}. \tag{5.10}$$

5.5.6.4 An example of the Kuhner move

Assume the ARG in Figure 5.9 (a) is G_j. First, branch B is randomly chosen amongst all the branches and is detached from the ARG. Detaching branch B results in the cancellation of the events at time t_2, t_3, and t_5. In Figure 5.9 (b), the state of the ARG after detaching B is shown. Branch B is now floating, and all its sites are eligible links (colored in green). Note that D, J, K, and L are the affected branches and have not yet been updated. The remaining steps are as follows:

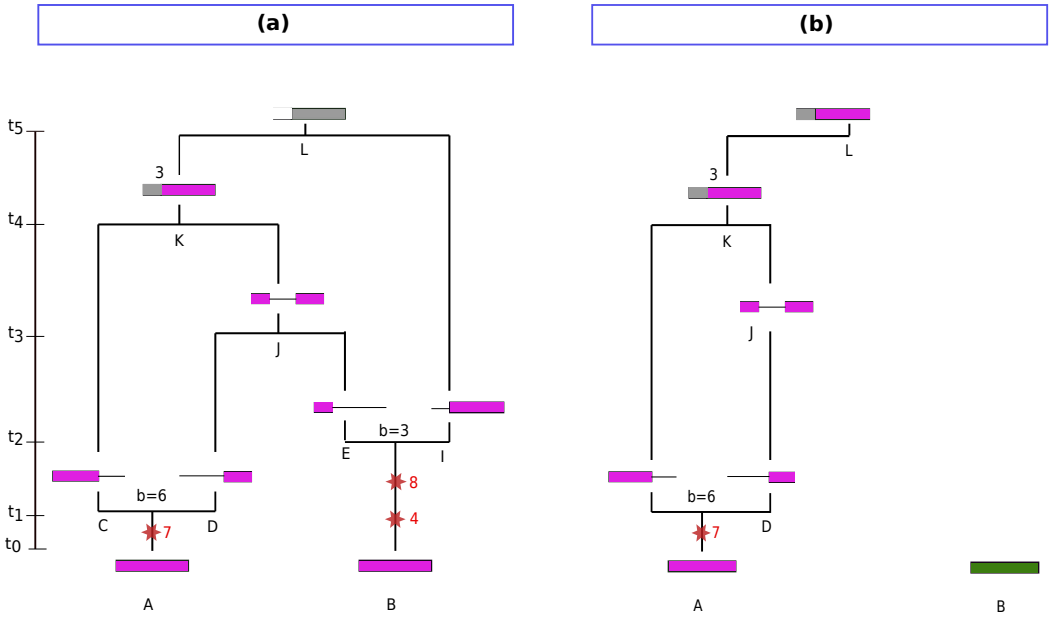


Figure 5.9: (a): An ARG of two sequences with length ten sites. ⋆: mutations with genomic location. (b): The ARG after branch B is detached.

- For the time interval (t_0, t_1), $k_i = 1$, $|F| = 1$, and $k'_e = 9$. A time point (t') is generated from an exponential distribution with rate

$$\frac{1}{2N} + 9r,$$

since $t' > t_1$, no new event occurs in this interval.

- The next time interval is (t_1, t_2) with $k_i = 2$, $|F| = 1$, and $k'_e = 9$. Thus, λ is

$$\frac{2}{2N} + 9r.$$

$t'_2 = t' + t_1 < t_2$, therefore a new event happens in this interval. The new event is a recombination chosen with probability $9r/\lambda$. A recombination breakpoint, say 4, is chosen at random among the eligible links on B and B is split into two parent branches, R and T.

- For the time interval (t'_2, t_2), $k_i = 2$, $|F| = 2$, $k'_e = 8$, and

$$\lambda = \frac{2}{2N} + 8r.$$

Since $t' > t_2 - t'_2$, no new event occurs in this time interval.

- For (t_2, t_3), no new event takes place. As seen in Figure 5.9 (b), the parent branch at t_3 is branch J and carries ancestral material on segments $[1, 2]$ and $[6, 10]$. In the proposed ARG, branch J does not exist and its path is followed by branch D that carries segment $[6, 10]$ as ancestral material. Since $[6, 10] \subseteq [1, 10]$, there is no eligible link on this branch in the forward

move. However, in the reverse transition, branch J is a partially floating lineage with eligible links on the interval $[1, 6]$.

- The next time interval is (t_3, t_4). Here, $k_i = 2$, $|F| = 2$, and $k'_e = 8$. The next event time (t') is generated from an exponential distribution with rate

$$\lambda = \frac{5}{2N} + 8r,$$

and $t'_3 = t_3 + t' < t_4$. The new event is a CA (probability $4/2N\lambda$). Branches R and C are chosen with probability $1/6$, and are coalesced at t'_3.

- For the time interval (t'_3, t_4), $k_i = 2$, $|F| = 1$, and $k'_e = 6$. No new event occurs in this time interval (Figure 5.10 (c)). The ancestral segment carried by branch K (at t_4) changes from $[3, 10]$ to $[4, 10]$ as a result of the CA event at t'_3. Hence, branch K will be partially floating in the reverse transition with eligible links on the interval $[3, 3]$.

- The next time interval is (t_4, t_5) where $k_i = 2$, $|F| = 1$, and $k'_e = 6$. No new event takes place in this interval. Note that the event at t_5 does not exist in G_{j+1}. However, we need to consider t_5 in our move because the number of eligible links may differ before and after t_5. The current ARG is shown in Figure 5.10 *(c)*.

- The last time interval is (t_5, ∞). Branch K is turned to a floating lineage. Hence, $k_i = 0$, $|F| = 2$, and $k'_e = 12$. A new time point (t') is generated from an exponential distribution with rate

$$\frac{1}{2N} + 12r.$$

The new event is a CA between branches K and T. This terminates the algorithm and the proposed ARG is shown in Figure 5.10 (d).

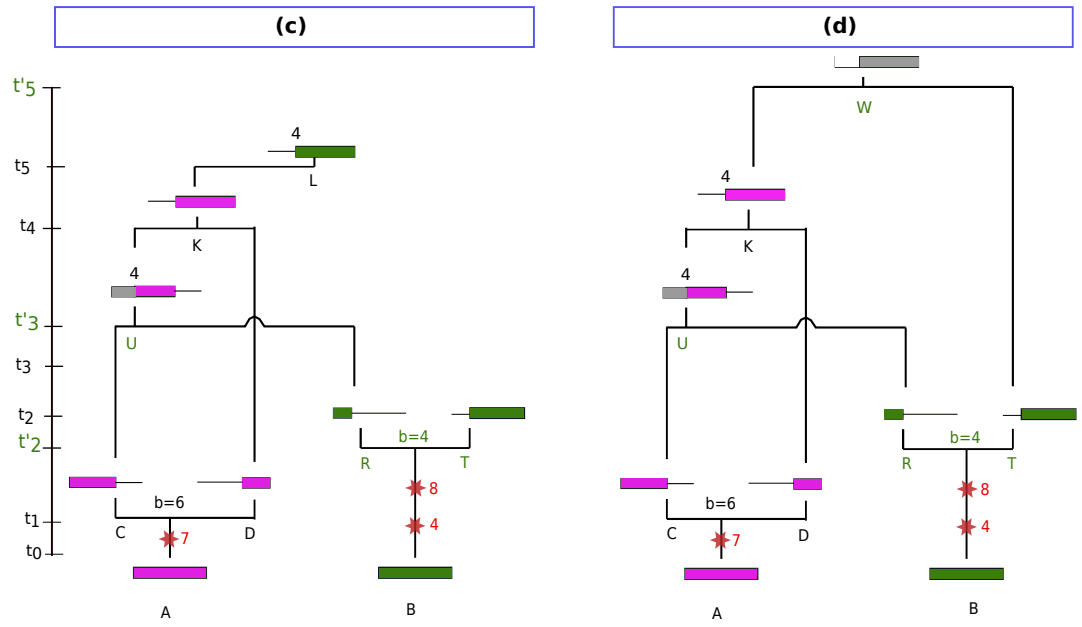


Figure 5.10: Panel (c): The state of the ARG after time interval (t_4, t_5). Panel (d): The ARG proposed by the Kuhner move.

We implemented the MCMC algorithm in a computer program called *ARGinfer*.

5.6 Discussion

In this chapter, we developed a new data structure (*ATS*), which augments TS to record an ARG and the mutation information. An *ATS* with mutations facilitates the MCMC algorithm in discovering incompatible proposals immediately after they are proposed. Therefore, unlike *FTS*, this data structure does not waste computation on building incompatible proposals. Another advantage of an ARG with mutations is that the likelihood evaluation does not require D. As discussed in the previous chapter, comparing DNA sequences against each other at each MCMC iteration is costly. With the *ATS*, the likelihood evaluation takes advantage of the mutation information incorporated in the data structure. The *ATS* is designed so that the MRCA relocations are straightforward to handle; the *ATS* includes the descendent samples for a segment that helps to recognize MRCAs without recording them.

We also introduced an MCMC algorithm to infer the ARG under the CwR using *ATS*, which is implemented in the software *ARGinfer*. Six proposals are developed to update the ARG. The sixth proposal is the Kuhner move, a modified version of the transition introduced in Kuhner et al. (2000). This transition resamples a part of an ARG under the CwR and allows to jump between dimensions with more than one difference in the number of recombination events. Hence, all the Kuhner proposals are *valid*.

The *ATS* eases the deficiencies of the *FTS*. Therefore, we expect that *ARGinfer* has better mixing and convergence powers than the MCMC algorithm in the

previous chapter. In the next chapter, we evaluate the performance of *ARGinfer*.

Chapter 6

Method evaluation

In this chapter, we assess the power of *ARGinfer* in inferring the ARG using simulated data. In section 6.2, we introduce some heuristics to monitor convergence of the MCMC chain. Section 6.3 compares *ARGinfer* against *ARGweaver* in estimating ARG parameters. In section 6.4, we discuss the performance of *ARGinfer* on real human data.

6.1 Simulated data

We used *msprime* to simulate a number of data sets with various settings. We set $\mu = 1 \times 10^{-8}$ and $N = 5000$. For r, we used three values,

$$r = 1 \times 10^{-8}, 0.5 \times 10^{-8}, 0.25 \times 10^{-8},$$

to assess the performance under different mutation to recombination rate ratios, i.e., $R = \mu/r = 1, 2, 4$. We assume that μ and r are constant along the genome. For each R, we generated 150 data sets, each with 10 sequences of length 10^5 sites under the CwR.

We applied *ARGinfer* to each simulated data set with 2×10^6 sampling iterations, of which 1×10^6 are discarded as burn-in, and every 500th sample is retained, resulting in 2×10^3 samples.

Similarly, we applied *ARGweaver* to the same data sets with 2×10^4 sampling iterations, of which 2×10^3 are discarded as burn-in, and every 10th sample is retained, resulting in 1.8×10^3 samples. We used the default value (20) for the number of discrete times. All other parameters were set to the default.

We ran the algorithms on Spartan, a High Performance Computing System operated by Research Platform Services at The University of Melbourne (Lafayette et al., 2016), with one Xeon(R) Gold 6154 CPU (1 core) and 15 GB RAM for each data set. Table 6.1 presents the running time for *ARGinfer* and *ARGweaver*. For $R = 1$, *ARGinfer* takes 19 hours on average to complete 2×10^6 MCMC iterations and *ARGweaver* spends about 5 hours to complete 2×10^4 iterations. For higher R, running time decreases for both algorithms, which is expected because there are fewer recombinations in the ARG, so the algorithm deals with smaller ARGs. In general, *ARGinfer* is slower than *ARGweaver*.

To assess whether 2×10^6 iterations by *ARGinfer* represent comparable outputs to 2×10^4 iterations by *ARGweaver*, we calculated the ESS (see section 4.4) for various ARG parameters. As seen in Table 6.2, the ESS values for the total branch length are comparable. However, the ESS for the posterior and total

Table 6.1: The running time for *ARGinfer* and *ARGweaver*. Here and in subsequent tables, red indicates the better result.

		CPU time (hour)		
Method	**MCMC iterations**	$R = 1$	$R = 2$	$R = 4$
ARGinfer	2×10^6	19	6.5	3
ARGweaver	2×10^4	5	4	3

Table 6.2: The average ESS of the total branch length, posterior, and total number of recombinations for various R from 150 data sets.

		ESS		
R	**Method**	**Branch length**	**Posterior**	**Total number of recombinations**
1	ARGinfer	327	249	248
	ARGweaver	383	585	708
2	ARGinfer	898	450	448
	ARGweaver	851	1225	1337
4	ARGinfer	1548	832	825
	ARGweaver	1190	1595	1628

number of recombinations is higher for *ARGweaver*. However, since our main focus is on comparing ARG parameters related to branch length and event times, we proceed with this setting.

The acceptance probability for *ARGinfer* is 0.3 on average. The transition types 1 to 6 (see section 5.5) are chosen with probabilities $1/14, 1/14, 1/14, 5/14, 1/14,$ and $5/14$, respectively. We gave a higher chance to the time adjustment and the Kuhner move because the former is the only transition type that resamples the event times, and the latter introduces the biggest change to the ARG. It is worth noting that we attempted various settings and experimentally concluded that the current setting has a more reasonable acceptance rate.

6.2 Convergence diagnostics

We assessed convergence through four ARG properties, the posterior density, total branch length, number of ancestral recombinations, and number of non-ancestral recombinations. The second and third properties represent the distribution of tree topologies and event times. The last property represents the distribution of the amount of TNAM in ARGs.

We used three heuristics to determine the validity of the MCMC algorithm. One way to assess convergence is to run the MCMC algorithm more than once with various initial ARGs and check if all the runs converge to the same distribution. The trace plots of various ARG features for two independent runs of a randomly selected data set with $R = 2$ are illustrated in Figure 6.1. As seen from the Figure, the chains stabilize around similar values for both runs for all the four features. We used two sided t-tests to examine whether the

means of the outputs are significantly different. The p-values for the posterior, total branch length, number of ancestral recombinations, and number of non-ancestral recombinations are $0.61, 0.89, 0.78$, and 0.45, respectively, indicating that both chains converge to the same distribution.

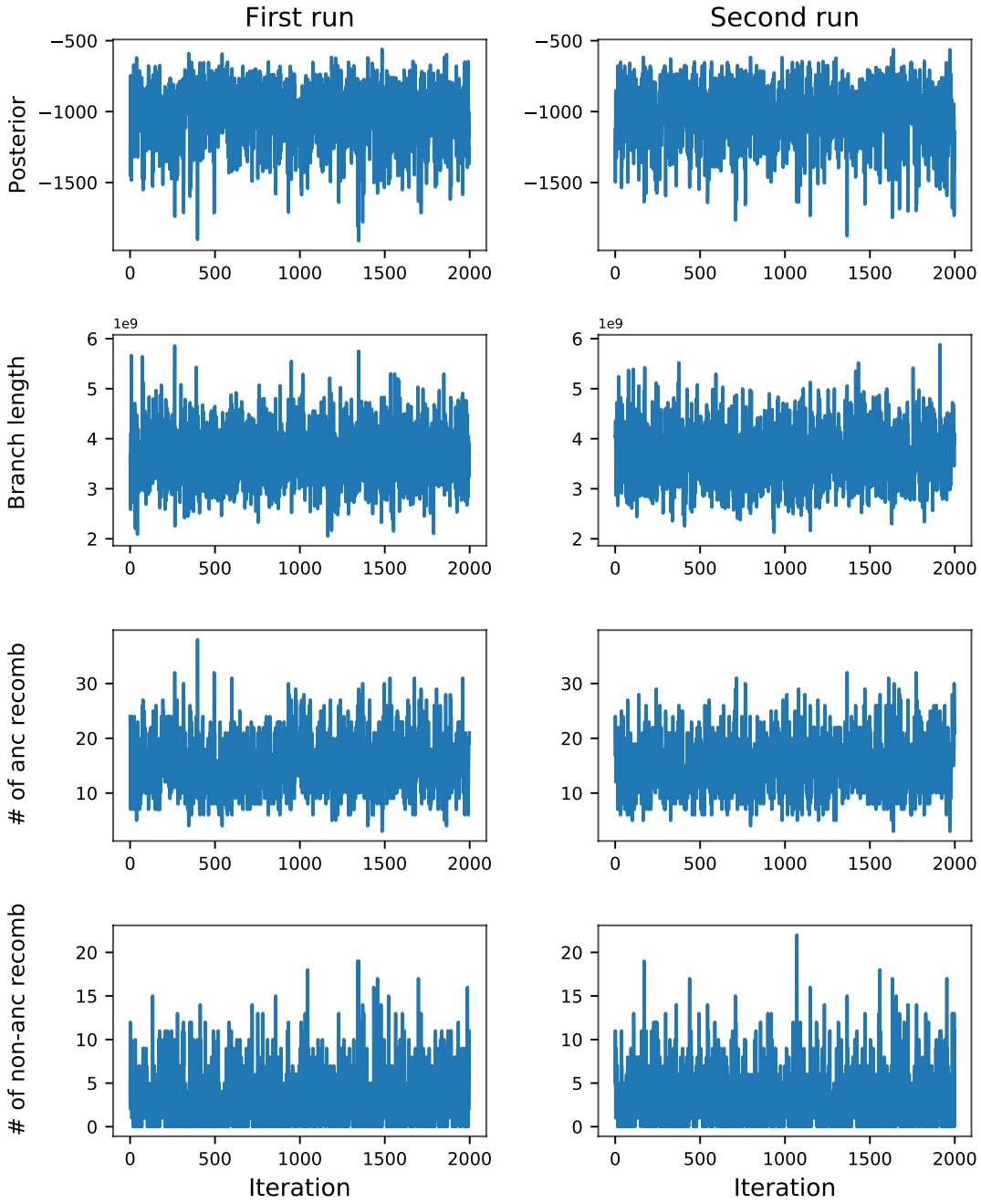


Figure 6.1: Trace plots of the posterior density, total branch length, number of ancestral recombinations, and number of non-ancestral recombinations for two independent runs of a randomly selected data set. The x axis is the MCMC iteration number for the 2×10^3 sampled ARGs.

We also examined the autocorrelation of the outputs for each parameter. From Figure 6.2, we observe that the autocorrelations decrease to < 0.3 for the

first lag and fall into the 95% confidence interval of no correlation after at most 3 lags.

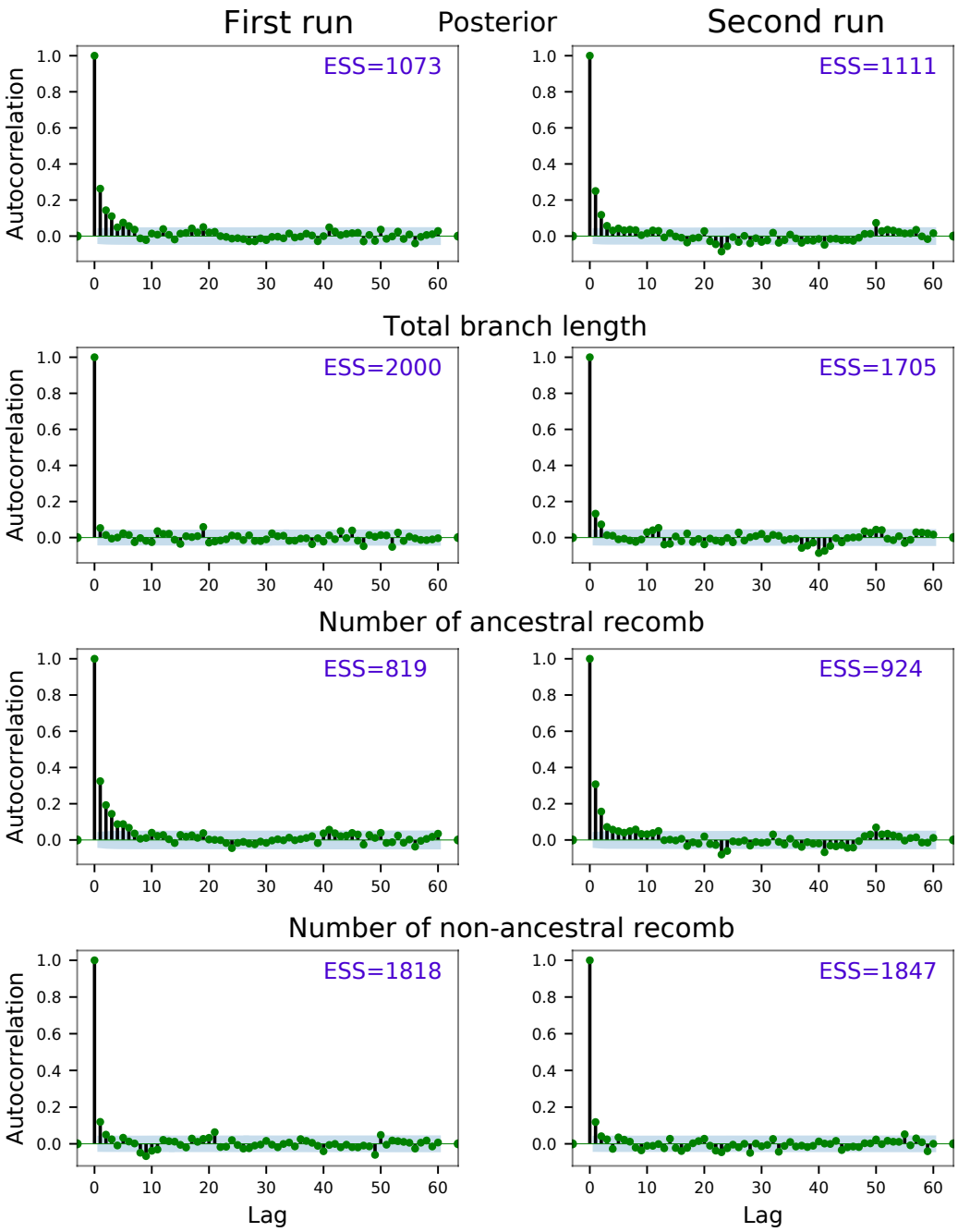


Figure 6.2: The autocorrelation in the posterior, the total branch length, the ancestral, and non-ancestral recombinations for the data set with trace plots in Figure 6.1. The blue shading represents the 95% confidence interval for no correlation.

From the trace plots in Figure 6.1 and the autocorrelation and ESS in Figure 6.2, we can conclude that these diagnostics do not indicate any non-convergence. We now proceed assuming convergence.

Table 6.3: The RMSE, 50% coverage, and average length of the 50% credible intervals of the inferred ARG total branch lengths per site (in generations) using *ARGinfer* and *ARGweaver* from 150 data sets for each of three values for R.

			50% credible interval	
R	**Method**	**RMSE**	**Coverage**	**Average length**
1	ARGinfer	6722	0.43	8240
	ARGweaver	6713	0.41	7885
2	ARGinfer	6248	0.52	8723
	ARGweaver	6430	0.52	8496
4	ARGinfer	6847	0.51	8800
	ARGweaver	6931	0.51	8730

6.3 Comparison study

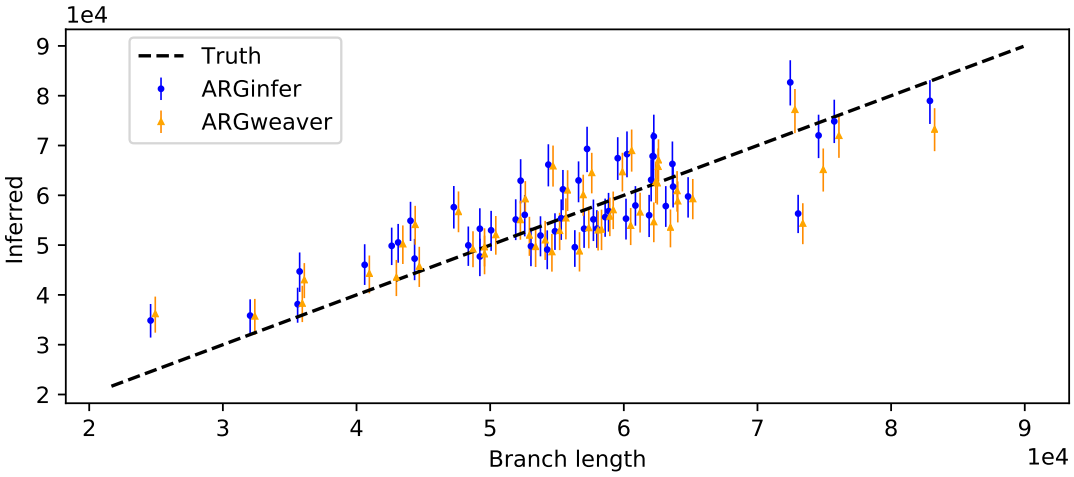
We compared *ARGinfer* and *ARGweaver* in estimating ARG parameters. The outputs are compared against the true ARGs that were simulated from *msprime*. It is worth noting that the target distributions of the algorithms are the CwR+ and the DSMC+, respectively, for *ARGinfer* and *ARGweaver*, but the true ARG is only a single realization of the target distribution, and it may or may not be a likely realization.

6.3.1 Total branch length

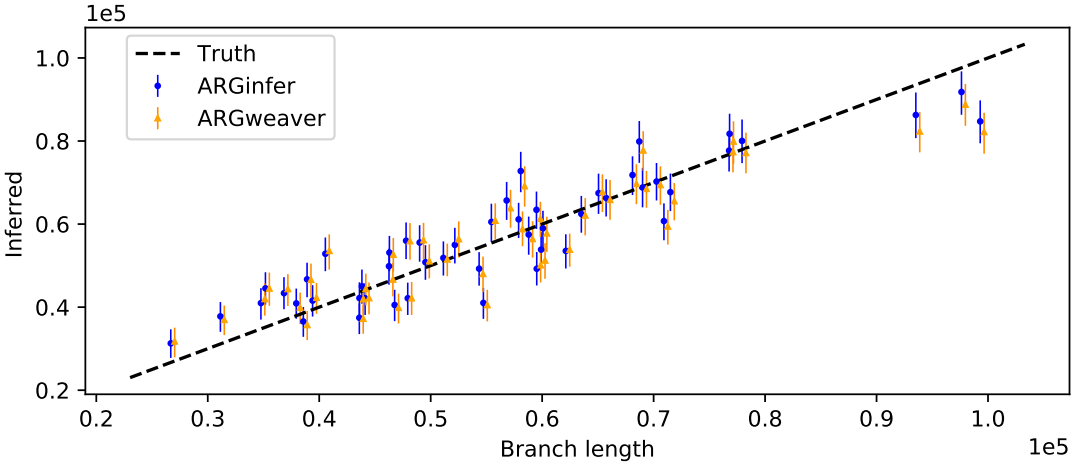
We investigated the ability of the algorithms to estimate the total branch length of ARGs. Figure 6.3 shows the true (dashed black line) versus the inferred total branch length per site in generations for *ARGinfer* and *ARGweaver*. For each $R = \{1, 2, 4\}$, 50 simulated data sets are randomly chosen from the 150 simulated data sets. Each point indicates the posterior expected value, which is the average over all 2×10^3 (1.8×10^3 for *ARGweaver*) sampled ARGs for a single dataset, and the error bars represent 50% credible intervals. We observe from the Figure that the inferred values for both methods are similar and close to the truth. However, for higher values (the right tail), *ARGinfer* tends to estimate higher values than *ARGweaver*. This difference is large for $R = 1$ (panel (a)) and decreases as R increases. This trend may relate to the amount of TNAM in the ARGs, which is higher for lower R.

Table 6.3 provides statistics for the estimated total branch length in all 150 data sets. The performances of both methods are similar, *ARGweaver* infers shorter 50% credible intervals than *ARGinfer* with only a small loss of performance in root mean square errors (RMSE) and 50% coverage.

(a) $R = 1$



(b) $R = 2$



(c) $R = 4$

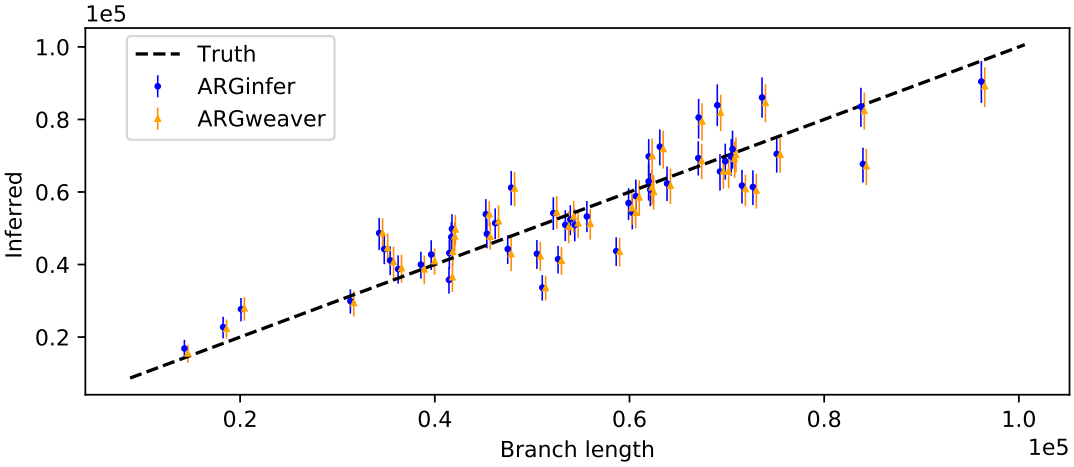


Figure 6.3: True versus inferred total branch length of 50 randomly selected data sets among the 150 simulated data sets. The vertical line segments are 50% credible intervals.

Table 6.4: Expected number of ancestral recombinations conditioning on $r = \{0.25, 0.5, 1\} \times 10^{-8}$. The values are calculated by $\rho \sum_{i=1}^{n-1} \frac{1}{i}$, introduced by Hudson and Kaplan (1985).

$R = 1$	$R = 2$	$R = 4$
57	28	14

Table 6.5: Statistics for the number of ancestral recombination events.

			50% credible interval	
R	**Method**	**RMSE**	**Coverage**	**Average length**
1	ARGinfer	9.30	0.48	11.90
	ARGweaver	9.53	0.54	11.73
2	ARGinfer	4.39	0.57	7.51
	ARGweaver	4.70	0.49	7.35
4	ARGinfer	3.70	0.52	4.76
	ARGweaver	3.83	0.52	4.67

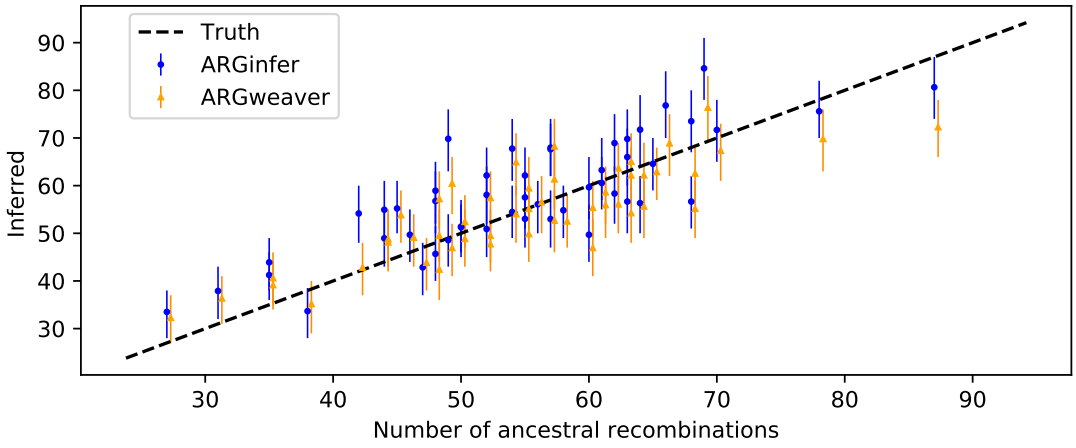
6.3.2 Number of ancestral recombination events

ARGinfer infers both ancestral and non-ancestral recombinations. On the other hand, *ARGweaver* can only infer ancestral recombinations. We can thus only compare the ability of the algorithms in estimating the number of ancestral recombination events. From the statistics reported in Table 6.5, *ARGinfer* performs slightly better in terms of the RMSE for all ratios. On the other hand, *ARGweaver* records a smaller average length of 50% credible intervals. The 50% coverages for both methods are similar and close to 0.50, indicating that the posterior distributions from both methods are well-calibrated.

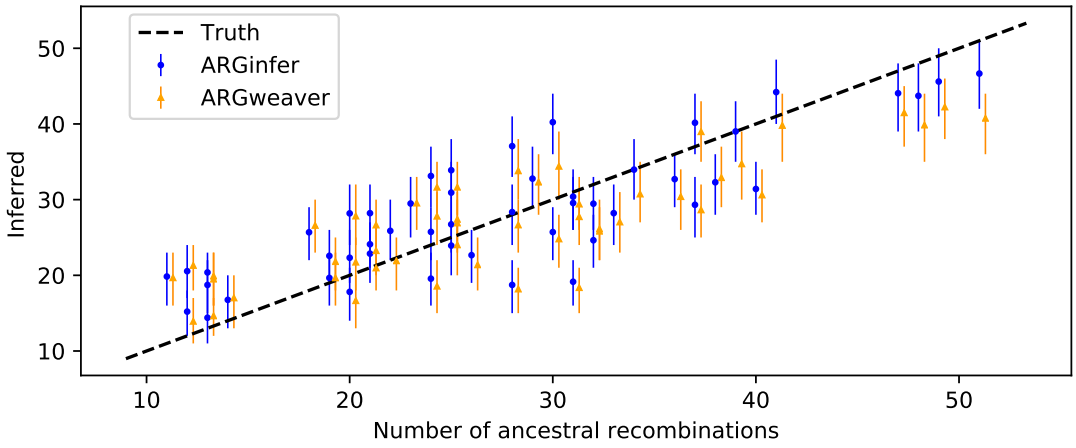
As seen from Figure 6.4, both methods show a consistent overestimation of low values and underestimation of high values for all R values. This deviation is consistent with *ARGweaver* results reported in Rasmussen et al. (2014), where the authors suggested that it is due to model misspecification of the DSMC. However, we suggest that it may be a shrinkage effect due to prior. Both methods assume that ρ is known, affecting estimation when the observed number of recombinations diverges from expected values. From Figure 6.4 and Table 6.4, we observe that the algorithms work well when the observed number of recombination is close to the expected values and shrink the number for extreme values.

From the results in this section and section 6.3.1, both *ARGinfer* and *ARGweaver* perform well in estimating the total branch length and the number of ancestral recombination. However, there is a systematic difference between some estimated values for the two methods. *ARGinfer* estimates are closer to the truth (based on RMSE values in Tables 6.3 and 6.5). The difference may reflect model misspecification of the DSMC assumed by *ARGweaver*.

(a) $R = 1$



(b) $R = 2$



(c) $R = 4$

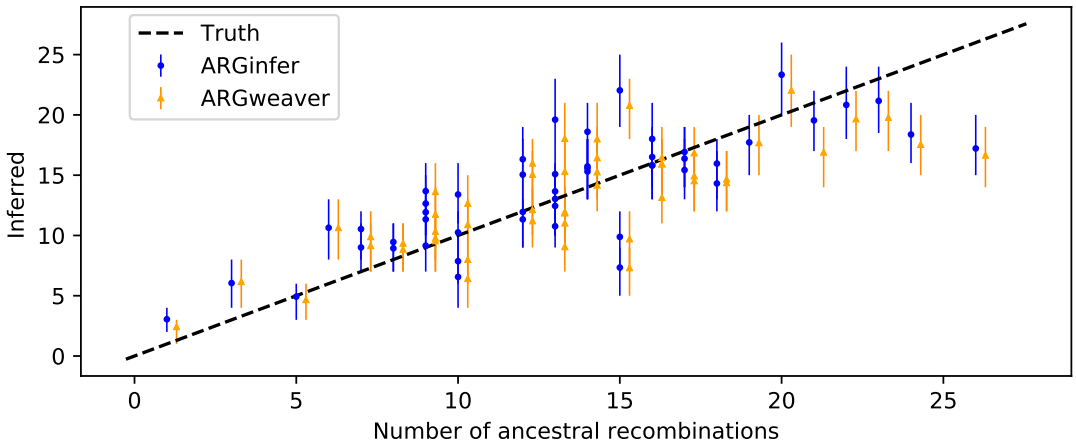


Figure 6.4: True versus inferred number of ancestral recombination events for the 50 data sets in Figure 6.3. The vertical line segments are 50% credible intervals.

6.3.3 Recombination rate

For a given ARG, the recombination rate per site per generation is the number of ancestral recombination events divided by the total branch length (in generations). Figure 6.5 provides the estimated recombination rates for 150 simulated data sets from *ARGinfer* and *ARGweaver* for $R = \{1, 2, 4\}$. For each scenario, we tested if the mean of the inferred recombination rates is equal to the true value using a one-sample t-test, and the p-values are reported.

From Figure 6.5, we can see that *ARGinfer* infers recombination rates with high accuracy for all the three ratios with non-significant p-values. On the other hand, *ARGweaver* underestimates the recombination rate significantly. For $R = \{1, 2\}$, there is a significant difference between the truth and the mean of the inferred values. For $R = 4$, the results are more accurate compared to the other ratios, but the p-value (0.0139) is still small. This smaller difference is because there is a higher number of mutation events, and consequently, the data is more informative about the ARG and the recombinations. The results indicate that *ARGweaver*'s estimates of the recombination rates are biased downwards.

We suggest that this superior performance by *ARGinfer* is due to modeling TNAM and non-ancestral recombination events. Unlike *ARGinfer*, *ARGweaver*

- ignores TNAM,
- does not allow "double-hit recombinations" (multiple recombinations at the same site),
- does not allow invisible recombinations, and
- discretizes event times.

From Table 6.6, we observe that the proportion of double-hit recombinations is negligible for all ratios. The proportion of invisible recombinations is higher but has an inverse relationship with the bias (Figure 6.5). Moreover, to investigate the effect of the number of discrete time points in the inference, we increased the number of time points for *ARGweaver* from 20 to 40 and re-ran *ARGweaver*. The p-values reported in Table 6.7 suggest that the estimates are slightly improved by doubling the number of time points. However, they continue to underestimate the recombination rate significantly. We, therefore, conclude that modeling TNAM improves the inference for the recombination rate.

Table 6.6: Proportion of invisible and double-hit recombinations in the sampled ARGs by *ARGinfer*. Each value is an average over all 150 data sets.

Recombination type	$R = 1$	$R = 2$	$R = 4$
Invisible	0.05	0.08	0.09
Double-hit	4.1×10^{-4}	1.7×10^{-4}	7.1×10^{-5}

Table 6.7: P-values for one-sample t-test of the inferred recombination rate by *ARGweaver* for 20 and 40 discrete times.

Number of discrete times	$R = 1$	$R = 2$	$R = 4$
20	6.8×10^{-20}	2.8×10^{-10}	0.014
40	5.03×10^{-14}	6.5×10^{-7}	0.05

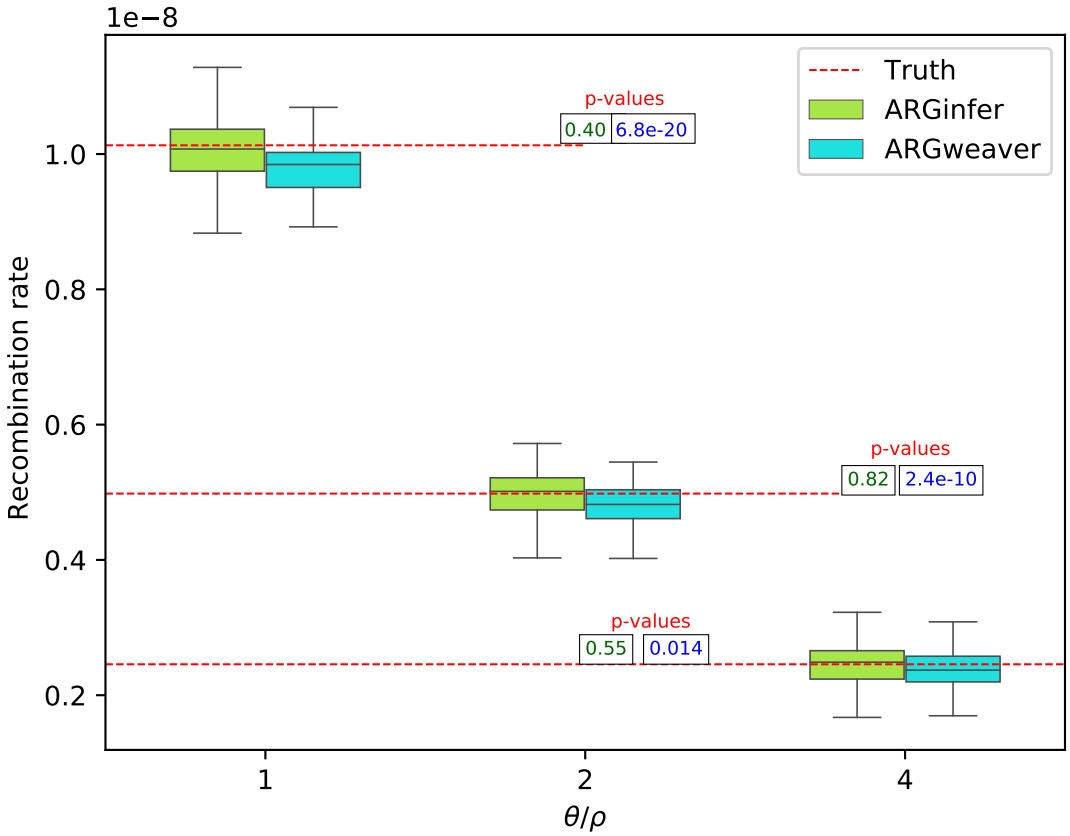


Figure 6.5: Estimated recombination rates using *ARGinfer* and *ARGweaver*. The truth is the mean of the true number of ancestral recombination events divided by the true total branch length.

Table 6.8: Statistics for TMRCA.

			50% credible interval		
R	**Model**	**RMSE**	**Coverage**	**Average length**	**Pearson coef.**
1	ARGinfer	7889	0.48	8816	0.54
	ARGweaver	7951	0.49	9473	0.53
2	ARGinfer	6369	0.51	7744	0.61
	ARGweaver	6513	0.51	8113	0.60
4	ARGinfer	5288	0.51	6334	0.55
	ARGweaver	5571	0.45	6386	0.51

6.3.4 TMRCA

As seen from Table 6.8, *ARGinfer* performs better than *ARGweaver* for all mutation-to-recombination ratios in estimating TMRCA along the genome. *ARGinfer* has a higher correlation (between the posterior mean and the true TMRCA at each genomic site), lower RMSE, and shorter length of the 50% credible intervals. One reason for this superior performance is that *ARGinfer* assume the CwR with continuous-time as prior, whereas *ARGweaver* assumes the DSMC. The impact of the time discretization adopted within *ARGweaver* can also be noticed from the jumps in the box-shaped 50% credible intervals in Figure 6.6, suggesting that accuracy is limited and depends on the pre-specified time points. We can also see that for some genomic intervals, the 0.25 and 0.75 quantiles are identical, which can be misleading in reporting uncertainty. On the other hand, the 50% credible intervals for *ARGinfer* are smoother and shorter (Table 6.8).

6.3.5 Allele age

We define allele age as the mid-point of the tree branch on which the mutation has taken place. In Table 6.9, we observe that *ARGinfer* outperforms *ARGweaver* in terms of all four statistics for $R = \{1, 2, 4\}$. *ARGinfer* has significantly smaller RMSE, shorter average length, and higher coverage of 50% credible intervals, suggesting that the branches on which the mutations are mapped are shorter than those of *ARGweaver*, and closer to the truth.

This large uncertainty for *ARGweaver* is due to the limitation of time discretization. By default, *ARGweaver* generates time points on a logarithmic scale so that recent time points are more close together than older points. Figure 6.7 shows that *ARGweaver* do not perform well for older mutations (particularly for $R = 2, 4$) because these branches are long due to less available time points and hence weaker estimates. One the other hand, *ARGinfer* do not limit the time points, which resulted in superior performance on both recent and old allege ages with smoother and shorter credible intervals (Figure 6.7).

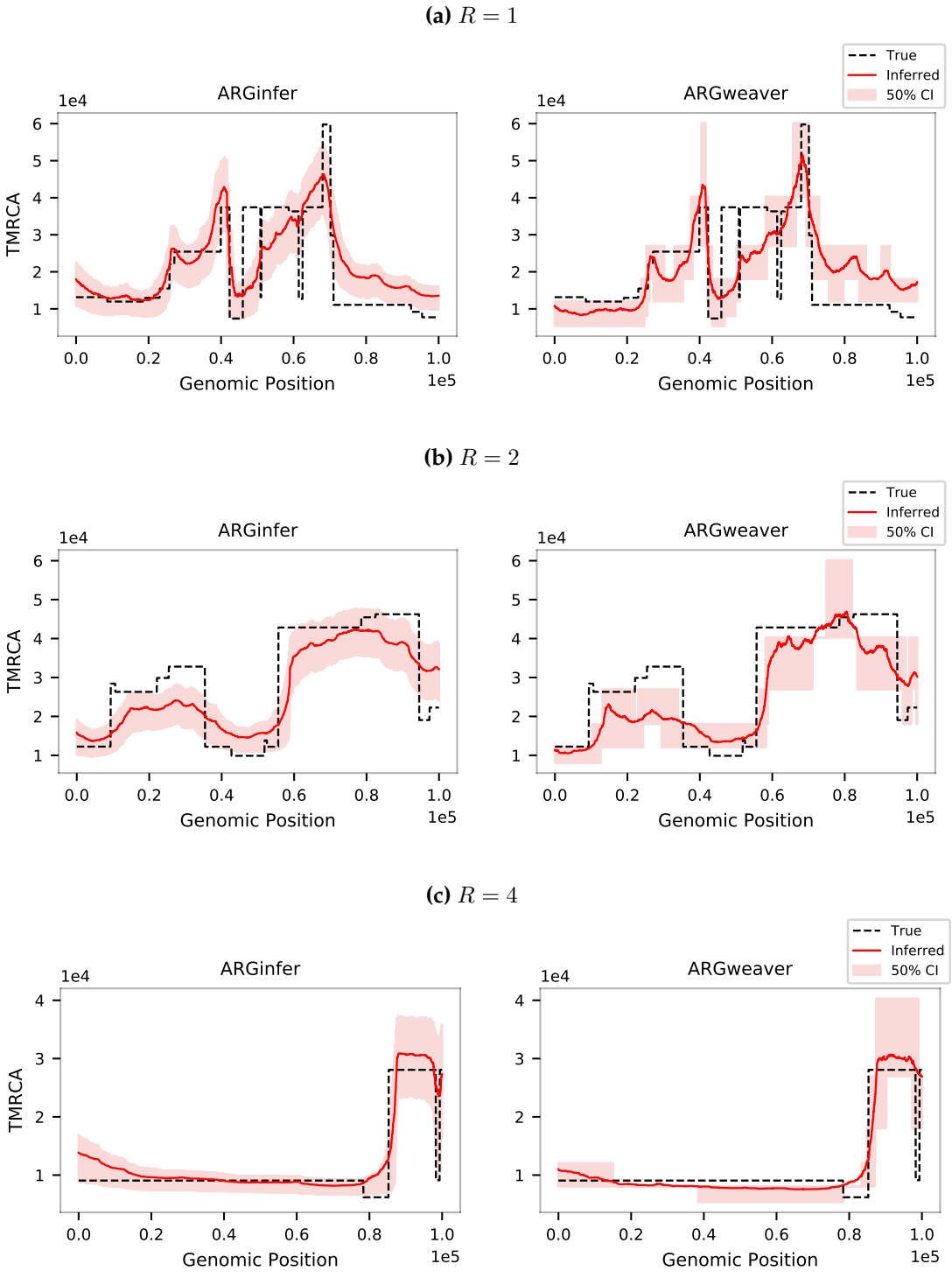


Figure 6.6: True versus posterior mean of TMRCA (in generations) by *ARGinfer* (left panel) and *ARGweaver* (right panel) for a randomly selected simulated data set. The black dashed line is the true TMRCA along the genome, the red line is the posterior mean TMRCA, and the red shade shows the 50% credible intervals.

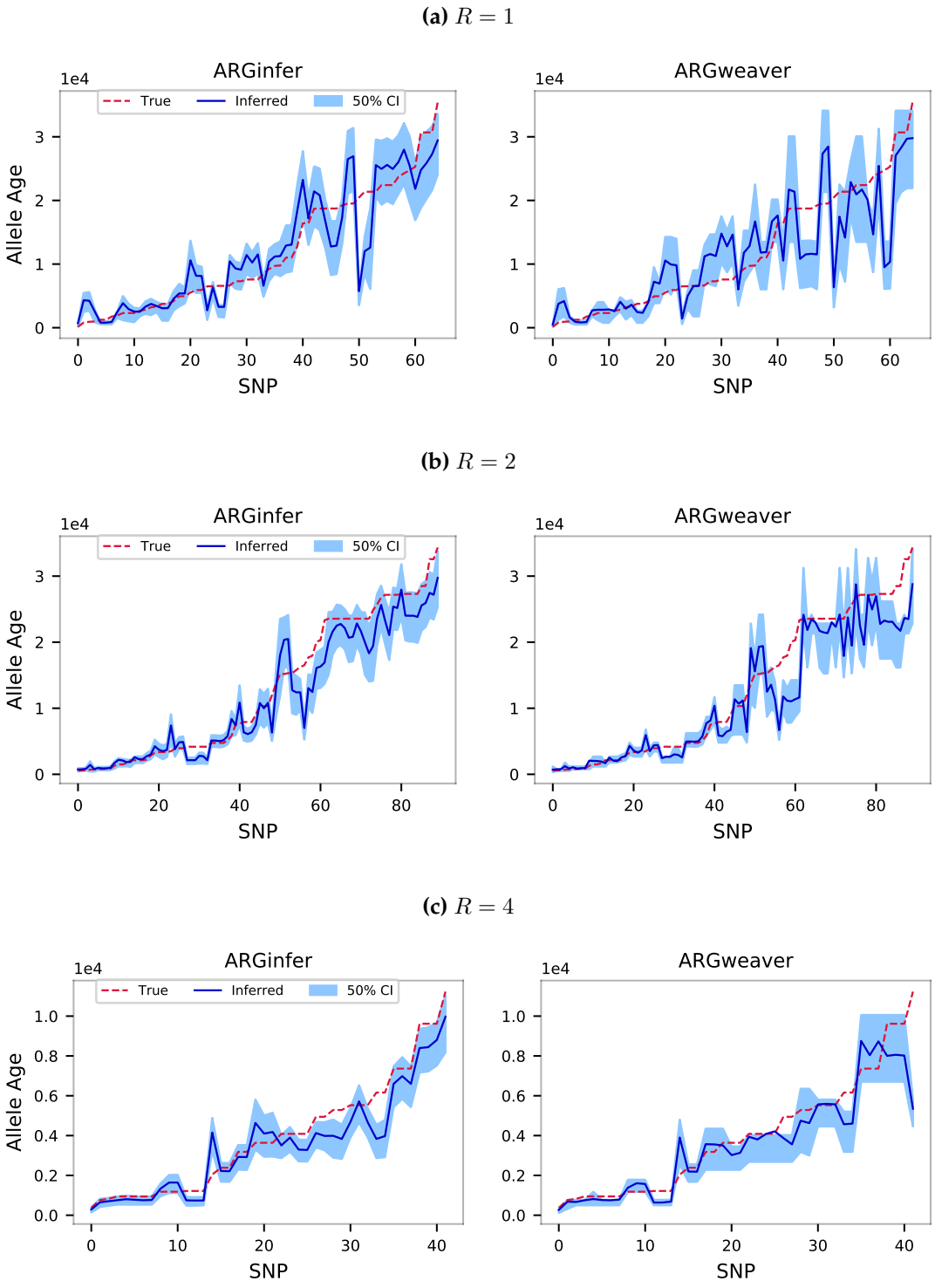


Figure 6.7: The posterior mean of allele ages (in generations) for the three random data sets in Figure 6.6. The x axis is ordered by increasing value of allele age for SPNs.

Table 6.9: Statistics for allele age.

			50% credible interval		
R	**Method**	**RMSE**	**Coverage**	**Average length**	**Pearson coef.**
1	ARGinfer	4086	0.47	4152	0.84
	ARGweaver	4697	0.49	4922	0.77
2	ARGinfer	3305	0.50	3498	0.90
	ARGweaver	3805	0.49	4075	0.85
4	ARGinfer	2534	0.50	2686	0.92
	ARGweaver	3109	0.46	3027	0.86

Table 6.10: Statistics for the total number of recombination events and posterior densities inferred by *ARGinfer*.

			50% credible interval	
R	**ARG feature**	**RMSE**	**Coverage**	**Average length**
1	Total recombination	17.87	0.44	20.61
	Posterior	526.02	0.43	605.10
2	Total recombination	7.70	0.52	10.44
	Posterior	230.21	0.50	311.30
4	Total recombination	4.57	0.56	5.64
	Posterior	137.42	0.49	170.05

6.3.6 Total number of recombinations and posterior density

We assessed the performance of *ARGinfer* in inferring the total number of recombination events (both ancestral and non-ancestral) and the posterior density. We calculated the true likelihood under the ISM, and the true prior by assuming the CwR. Note that *ARGweaver* assumes the Jukes-Cantor mutation model, and so because the likelihood evaluations are under different models, a posterior comparison is not meaningful.

It is seen from Figure 6.8 that *ARGinfer* performs well in estimating both the total number of recombinations and the posterior. As shown in Table 6.10, the 50% coverage for all the ratios is close to 0.50 for both features. As R increases, the RMSE and length of the 50% credible interval decrease, indicating that the method is more accurate, which is expected because the data is more informative about recombination events and rates. We observe a tendency to underestimate the number of recombinations on the right tail. The potential reason is discussed in section 6.3.2.

In conclusion, the results show that *ARGinfer* can successfully infer ARG parameters and is superior to *ARGweaver* in inferring the TMRCA, allele age, and recombination rate. We observed that the quality of inference depends on R. For $R = 1$, *ARGinfer* performs well but works better for higher R, i.e., lower recombination rate. This is because there is a smaller number of recombination events, resulting in higher information (larger number of mutation events) for tree inference in each non-recombining block.

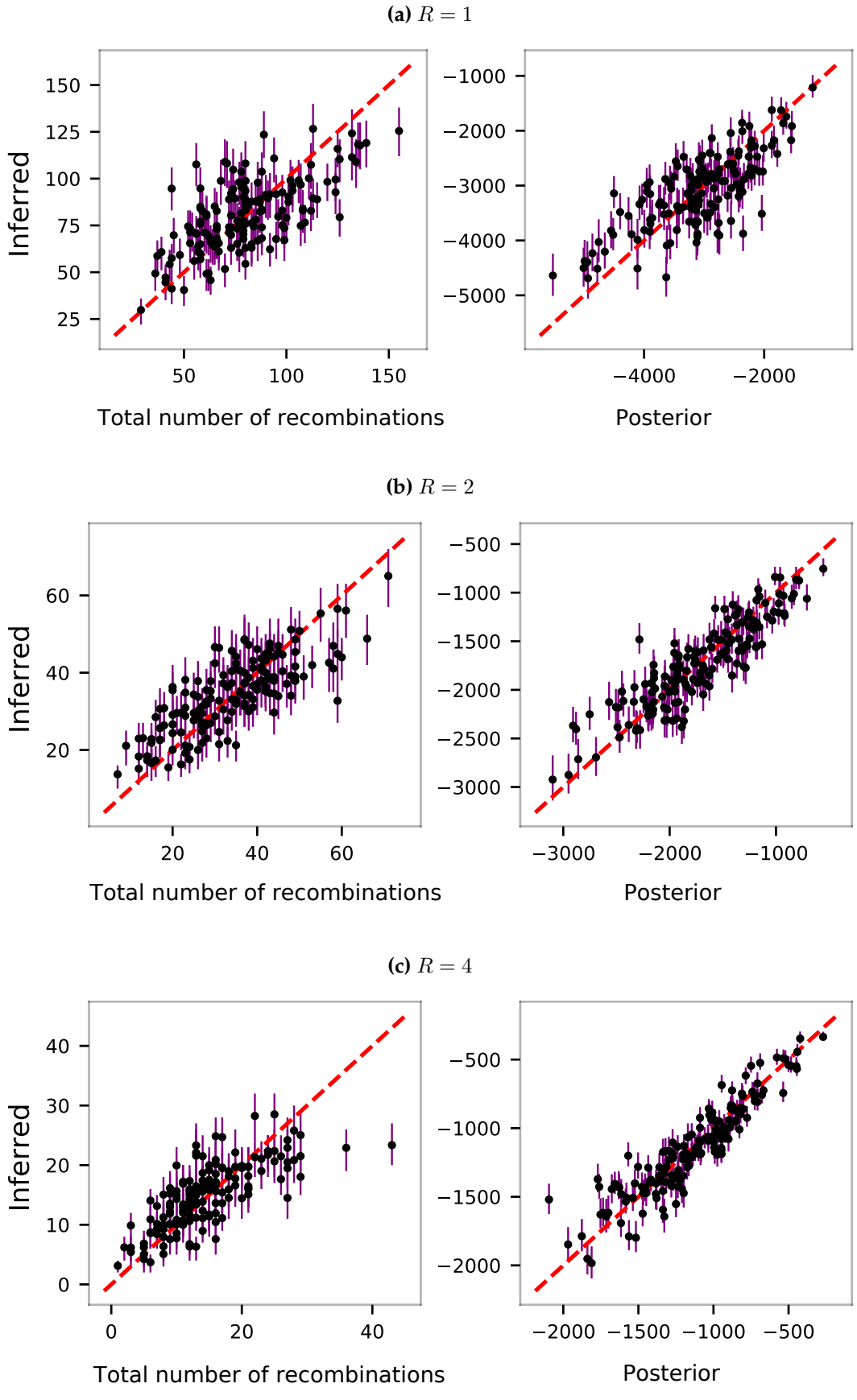


Figure 6.8: True versus *ARGinfer* posterior mean and 50% credible intervals for the total number of recombinations and posterior density.

6.4 Analysis of real data

We applied *ARGinfer* to real human genome sequences. We chose 5 unrelated African individuals from the ESAN (Nigeria) population from the 1000 Genome Project Phase III (1000 Genomes Project Consortium, 2015). We chose 5×10^4 sites from 1.8×10^6 site on chromosome 1. To minimize the influence of sequencing errors, we filtered the sequences using Plink (Purcell et al., 2007) such that:

- Minor Allele Frequency (MAF) > 0.11, i.e., the SNPs with minor allele count $= 1$ are excluded.
- Hardy-Weinberg Equilibrium (HWE) $> 10^{-5}$ to discard all SNPs which have HWE exact test p-value below $> 10^{-5}$.
- Only biallelic SNPs are kept.

The ancestral alleles were found from human genome dating (Albers and McVean, 2020), which provides the ancestral alleles based on related species. Following Rasmussen et al. (2014), we assume $\mu = r = 1.26 \times 10^{-8}$, and $N = 10^4$. We ran *ARGinfer* for 2×10^6 MCMC iterations and retained every 500th sample, after an 10^6 burn-in. The running procedure took 21 hours, and the acceptance probability was 0.2. We aimed to estimate the TMRCA, total branch length, recombination rate, and allele ages.

The trace plots of various ARG features for two independent runs are illustrated in Figures 6.9. As seen from the Figures, the chains are not stabilized around a value. From Figure 6.10, we observe that the autocorrelations are high and do not fall into the 95% confidence interval of no correlation even after lag 60. In addition, Figure 6.10 shows that the ESS is small for 2×10^3 outputs. These results suggest that convergence has not yet been reached.

We conclude that the current version of *ARGinfer* cannot handle real data, and further work is needed to improve the performance. Two potential reasons have been identified so far. The first reason is the existence of double-hit or back mutations in real data. Our algorithm assumes the ISM and does not allow such mutations, forcing the algorithm to generate more recombinations than the actual number. If a site requires a double-hit mutation to construct a compatible ARG with data, the algorithm may separate the site from the neighboring sites through recombination. As a result, depending on the number of back mutations in a data set, convergence can be slow. Another reason is the initial algorithm, which generates many recombination events for even small real data sets. Unlike simulated data, we observed that the initial algorithm introduces a large number of recombination events, approximately 10 times as many as the actual number. Many of the proposed CA events are rejected due to incompatibility and hence give a higher chance to recombination events, which occur on a randomly selected sequence. As a result, the initial ARG is a complex and less likely state, slowing down the algorithm exploring the ARG space.

Future plans

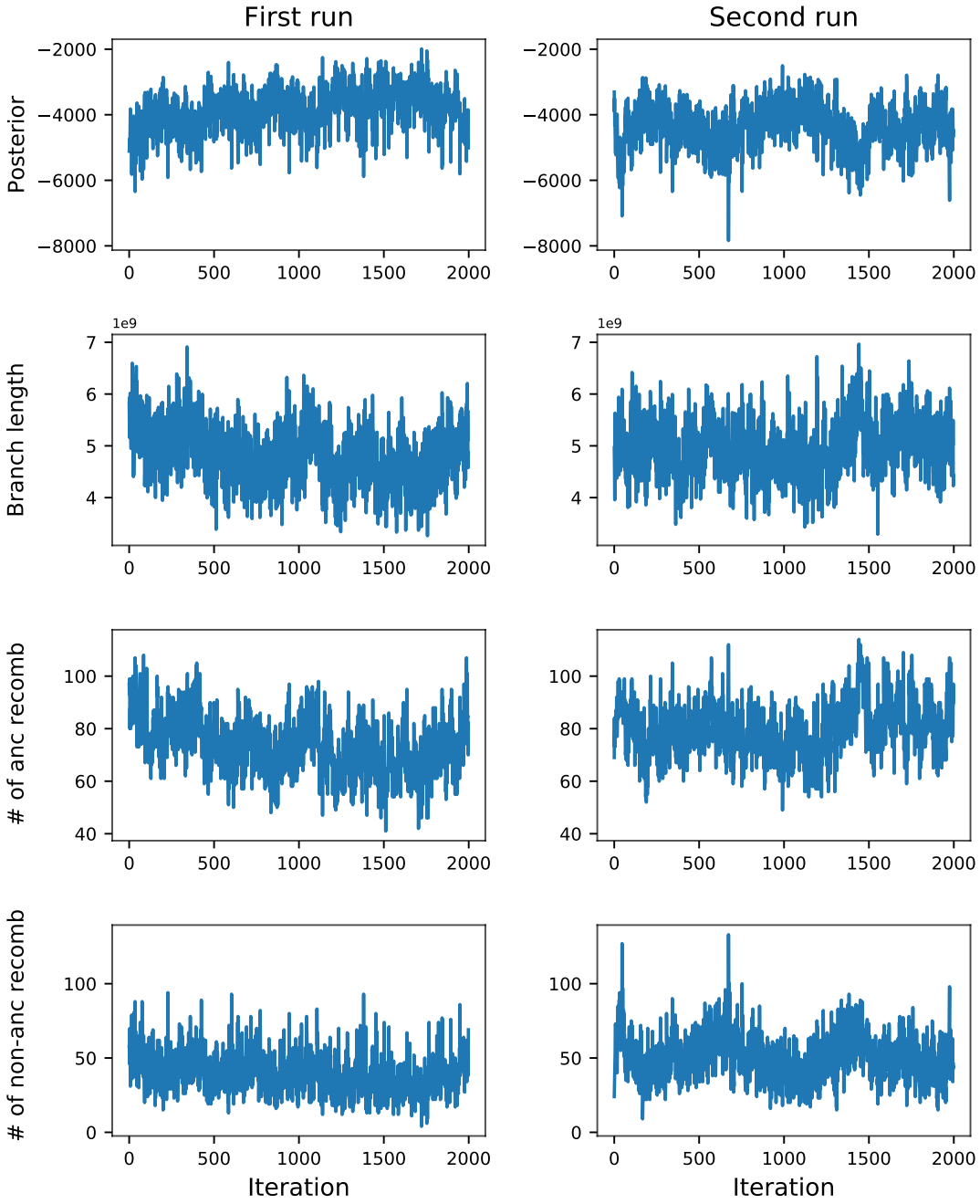


Figure 6.9: Trace plots of the posterior density, total branch length, number of ancestral recombinations, and number of non-ancestral recombinations for two independent runs on the same data. The x axis is the MCMC iteration number for the 2×10^3 sampled ARGs.

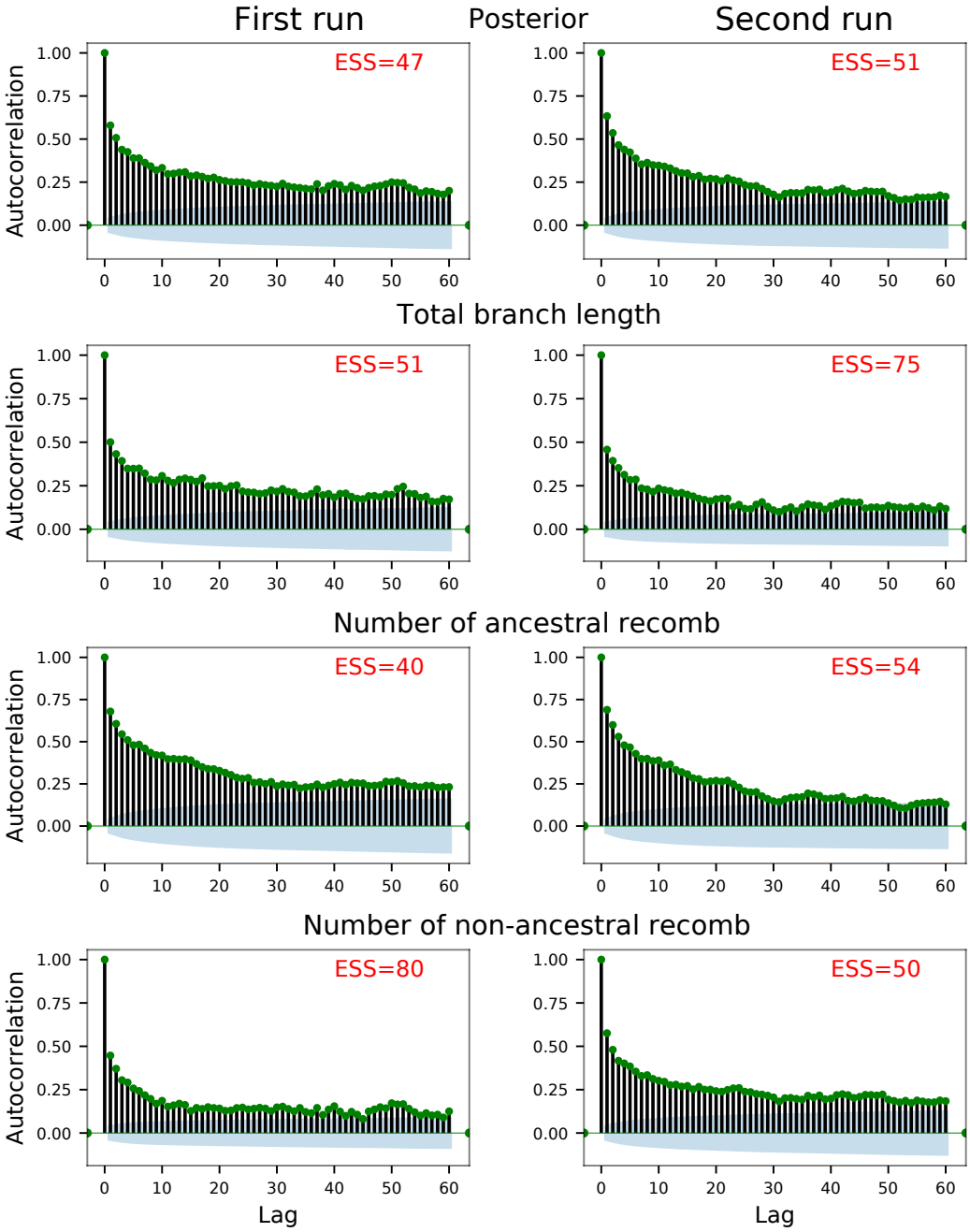


Figure 6.10: The autocorrelation in the posterior, the total branch length, and the number of ancestral and non-ancestral recombinations for the data set with trace plots in Figure 6.9. The blue shadow represents the 95% confidence interval for no correlation.

At the beginning of this research, our primary focus was on the MCMC algorithm and attempted to make it work under simple scenarios, such as the ISM. Now that we have enough evidence that the algorithm performs well, we could improve it by assuming a more realistic mutation model. Similarly to *ARGweaver*, we will incorporate the Jukes-Cantor model (called JC69), introduced by Jukes et al. (1969), to *ARGinfer* in the future to overcome the first limitation. JC69 is the simplest finite sites model that assumes that nucleotide substitutions occur at an equal rate and allows multiple mutations for a site.

JC69 is more realistic than the ISM but has greater computational cost. To evaluate the likelihood under the JC69, we need to integrate over all possible nucleotide transition events ($A \leftrightarrow T$ and $C \leftrightarrow G$) and transversion events (all other events) for the internal nodes. However, we can slightly modify the *ATS* to evaluate the likelihood more efficiently. We store the relevant likelihood on each branch and adjust them as the branches change while exploring the ARG space. We expect that this modification reduces the computation significantly since only a small portion of ARG branches are changed at each MCMC iteration.

The second limitation will also be alleviated by improving the current initial algorithm. We can modify the recombination and CA operations so that the sequences are not chosen uniformly. We can also adopt other existing heuristic algorithms, such as *tsinfer*, to generate a compatible and plausible ARG for real data sets. The difficulty of using *tsinfer* is that it does not provide any information about the real number of recombination and recombination breakpoints. More importantly, *tsinfer* may generate marginal trees that do not satisfy SPR operation.

6.5 Discussion

In this chapter, we evaluated the performance of *ARGinfer* in inferring ARG parameters using simulated data. We showed that the algorithm is capable of estimating the ARG parameters with high accuracy. *ARGinfer* improves the inference quality for the recombination rate, TMRCA, and allele ages, and is at least as accurate as *ARGweaver* for other parameters.

ARGinfer assumes a more realistic evolutionary model than *ARGweaver* and can infer all the recombination events that contribute to the structure of the observed data: ancestral, non-ancestral, and *invisible* recombinations.

Chapter 7

Conclusions and future directions

Inferring evolutionary history from a sample of DNA sequences is a long-standing problem in population genetics. The existing methods are either limited to small numbers of sequences or assume approximations of the CwR. In this thesis, I developed a statistical method to perform inference under the CwR. The main motivation of this work was the TS, which made efficient simulations of millions of genome-scale sequences under the CwR possible for the first time. Another motivation was the lack of a probabilistic model for sampling from the CwR+ (the posterior of ARGs, assuming the CwR as the prior distribution). After the SMC has been introduced, researchers have mostly been focusing on developing SMC-based methods. However, in chapter 3, we showed that the SMC ignores a large amount of information in the data. This research allowed us to explore the extent to which it is advantageous to assume the CwR rather than SMC when the data follows the CwR.

In section 7.1, I review the method I developed and summarize the ideas, findings, and conclusions. In section 7.2, I discuss potential future improvements. Section 7.3 provides a brief summary of the method.

7.1 Methodology overview and conclusions

We introduced an MCMC method, *ARGinfer*, that samples from the CwR+ given a sample of DNA sequences. The main innovation of *ARGinfer* is a novel data structure (*ATS*) that we designed to represent an ARG. As discussed in chapter 5, the *ATS* takes advantage of TS properties and stores all information in an ARG as well as mutations. We also introduced algorithms for constructing compatible ARGs from given sequences and evaluating the likelihood efficiently by incorporating the TS idea of no repetition in visiting the branches. Lastly, we developed simple and effective transition types to explore the ARG space, aiding the algorithm to converge to the CwR+ eventually. In chapter 6, we showed that *ARGinfer* mixes and converges well. In the following, I briefly review the main findings:

1. *ARGinfer* successfully samples from the CwR+, and thus provides an approximate posterior distribution over ARG-derived properties such as the ARG total branch length, number of ancestral and non-ancestral recombinations, allele ages, TMRCAs, and recombination rate.

2. *ARGinfer* improves inference accuracy. It is as accurate as of the state-of-the-art method for the total branch length and number of ancestral recombinations and is more accurate for the recombination rate, allele age, and TMRCA along the genome.

3. *ARGinfer*, for the first time, allowed us to investigate the importance of TNAM information in ARG inference. In chapter 6, we showed that *ARGinfer* can estimate the recombination rate with high accuracy. On the other hand, *ARGweaver*, which ignores TNAM information, underestimates the recombination rate and is biased. This difference between the estimates can be due to the invisible recombinations or (and) TNAM, which are not allowed in *ARGweaver* but exists in *ARGinfer*. The TNAM might affect the other estimations, but we could not explicitly quantify them. Further work is therefore needed to fully understand the impact of TNAM on the inferences and LD.

4. *ARGinfer* works well for both low and high recombination rates. When the recombination rates are low, *ARGinfer* converges quickly, but it becomes computationally expensive when the recombination rates are high.

5. Although *ARGinfer* cannot handle as many sequences as *ARGweaver*, it has better scaling properties than the existing methods under the CwR. The current version of *ARGinfer* can handle about 15 sequences with 1×10^5 to 5×10^5 sites, which is a vast improvement over the MCMC method introduced by Kuhner et al. (2000) that handles 10 sequences with 10^3 sites length. This efficiency gain is due to the following three reasons:

    - Since identical subtrees between consecutive trees are stored only once in the *ATS*, the likelihood evaluation algorithm does not waste computation on visiting them separately.
    - The mutation information in the *ATS* provides the information required for likelihood evaluation without comparing the sequences together site by site.
    - Incompatible moves are rejected immediately after they proposed, without costly likelihood evaluations.

7.1.1 Assumptions and limitations

ARGinfer does make several assumptions. First, we assume the ancestral and derived alleles are known. The ancestral alleles can be either inferred by the existing methods or be found by sequencing related species. Second, we assume the simplest version of the CwR, hence, no selection, no demographic and population structures. Third, we adopt an approximate version of the ISM, where each site has undergone at most one mutation during its evolution, and the number of sites on the genome is finite. Therefore, our method cannot handle sites with more than two states.

Although *ARGinfer* takes advantage of the efficient properties of the TS, it cannot handle large data sets. The proposal distribution is limited to small

rearrangements on the ARG to preserve reversibility, leading to poor mixing when the number of sequences grows large. Another limitation is incompatible proposals that grow with the number of sequences and decrease the MH acceptance rate.

7.2 Future developments

7.2.1 Improving SPR

Uniform sampling of the potential reattachment branches (Z) in the SPR move (see section 5.5.1) results in a low acceptance probability when the sample size is large. The move can be improved by sampling from a subset (Z_c) of Z that includes potential compatible reattachment branches. To illustrate, in Figure 7.1 *(b)*, branch E is pruned. The potential reattachment branches are $Z = \{D, B, C, H, I, J, K, L\}$. Under the ISM, the move is compatible only if one of the two branches B or J is chosen to coalesce with E. Hence, with probability $P_{c'} = 6/8$ the move is incompatible. We can reduce the probability of $P_{c'}$ by sampling from Z_c rather than Z. We define two methods to find Z_c:

1. **Conservative method ($P_{c'} = 0$)**: Assume d is the pruned branch, c is the sibling of d, and Z_c is an empty set. We start from c, and scan the branches upwards (toward the GMRCA), and add the visited branches to Z_c until a relevant mutation (a mutation at s is relevant if $s \in S_d$, see Table 5.2), or a recombination event is encountered. Then we traverse the branch downward (toward the leaves) and stop if a relevant mutation or recombination is encountered. For instance, for the ARG in Figure 7.1 *b*, we start from B and move up the ARG until we reach the recombination event at t_4. B is a leaf, and so no downward traversal is possible; hence, $Z_c = \{B\}$. This move limits the SPR rearrangement strictly, but all the proposed moves are guaranteed to be compatible.

2. **Less conservative method ($P_{c'} > 0$)**: We start from the sibling branch, and traverse the ARG until a relevant mutation is encountered. Unlike the conservative method, we can pass through the recombination events and assume both parents are allowed. This move occasionally proposes incompatible ARGs, which can be identified in the SPR update step. For example, in Figure 7.1 *b*, we get $Z_c = \{B, I, L, K, J\}$, which reduces $P_{c'}$ from 0.75 to 0.375. Note that all the excluded branches (D, C, H) are incompatible with probability 1.

7.2.2 Estimating evolutionary parameters

The evolutionary parameters $\Theta = (\mu, r, N)$ are assumed to be fixed in the current version of *ARGinfer*. In section 6.3.2, we discussed that this assumption affects the estimation of the number of recombinations both in *ARGinfer* and *ARGweaver*. The algorithm can be extended by also performing MH moves in the parameter space of Θ. We can assume a Gaussian prior for Θ and add a

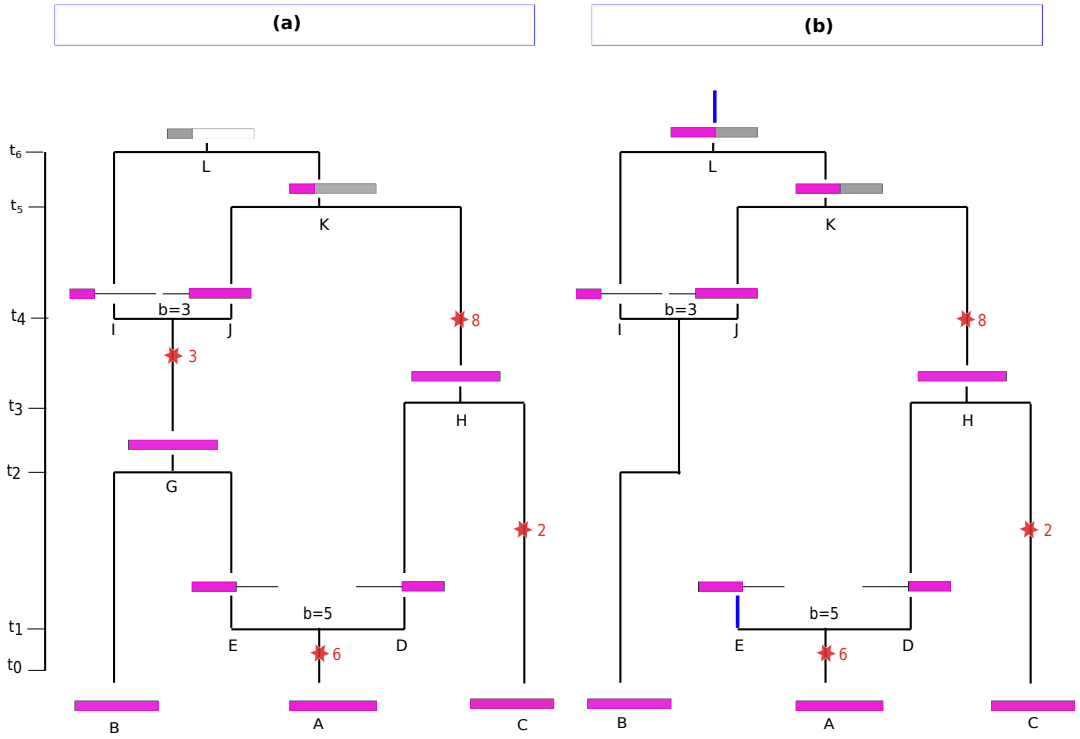


Figure 7.1: This is a copy of Figure 5.6. *(a)*: The parent p is not a root. *(b)*: p is a root. Blue branches are floating lineages.

new transition move to *ARGinfer*. At each iteration, new values for μ, r, and N can be proposed from a normal distribution centered at the current values and a constant variance.

7.2.3 Gene conversion

Gene conversion is an important factor that influences the evolutionary history of sequences. It transfers a short segment of ancestral material from one sequence without an exchange of flanking segments. Figure 7.2 shows a gene conversion backward in time. Although indirectly, *ARGinfer* includes the effect of gene conversion events by allowing them to be constructed by two recombination events and one CA event (Figure 7.2 *b*). *ARGinfer* can be extended to model gene conversion by assuming the coalescent with gene conversion process, introduced by (Wiuf and Hein, 2000). That is, going backward in time, a sequence can either experience a CA or a gene conversion event (that also includes recombination). This assumption introduces complexities in recombination and gene convergence rate and needs further investigation. Note that the SMC-based methods entirely ignore such an effect.

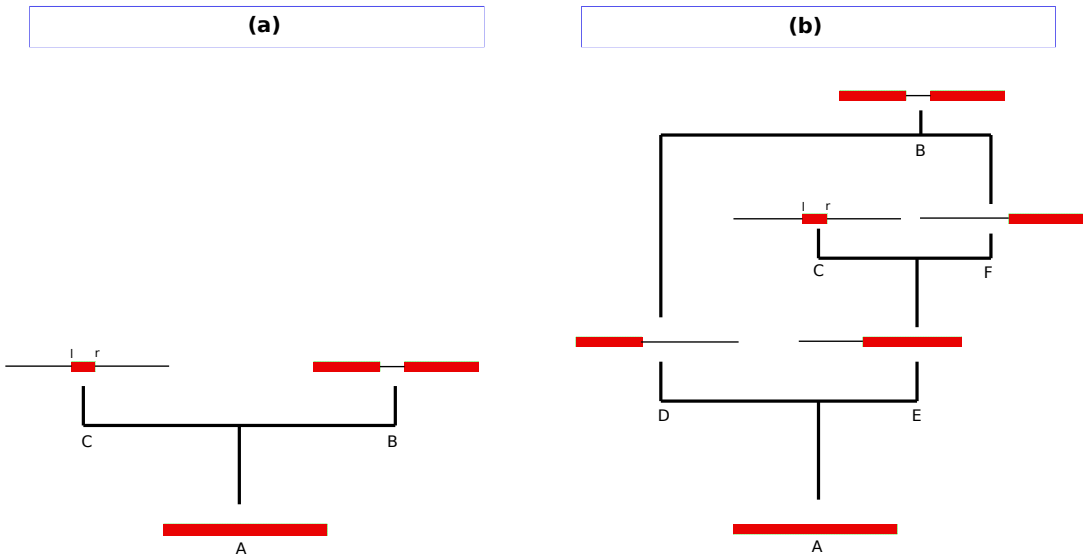


Figure 7.2: *(a):* A gene conversion on sequence A resulting in two sequences B and C. *(b)*: The topological effect of the gene conversion event in *(a)* is obtained by two recombination events and a CA event.

7.2.4 Non-reversible MCMC methods

Reversibility is a major limitation in *ARGinfer* and, in general, reversible MCMC algorithms. First, rearrangement of branches can create NAM lineages that result in irreversibility if it cancels a recombination event. This condition restricts proposals to small local rearrangement on the ARG. Second, storing these NAM lineages is an expensive task and complicates the proposal distribution considerably. Substantial improvement can be made by developing non-reversible sampling methods to sample from the posterior distribution. For instance, the zag-zag process, a continuous-time MCMC method, which is rejection-free and non-reversible, can be applied.

Koskela (2020) constructed a zig-zag process for inference under the standard coalescent. The zig-zag process is constructed such that continuous branch lengths and discrete tree topologies are embedded into a single continuous space, following the Probabilistic Path Hamiltonian Monte Carlo technique (Dinh et al., 2017). Koskela (2020) demonstrated that the zig-zag process is more efficient than MH under the ISM in terms of effective sample size. The zig-zag process is an important path forward from the MH toward the feasible Hamiltonian Monte Carlo methods and can be extended to include recombination. However, the feasibility and usefulness of this approach has yet to be fully studied.

7.2.5 *ARGinfer* and approximations

- **tsinfer:** It would be possible for *ARGinfer* to begin with *tsinfer*-based TS and resample the old events, keeping the recent events fixed. The magnitudes of errors incurred in *tsinfer* are higher for older events than

the more recent events because the former may never be strongly constrained by the data (Wakeley, 2020). Since most of the events in an ARG are recent, it is expected that this combination of *ARGinfer* and *tsinfer* improves scalability in *ARGinfer* and accuracy in *tsinfer*.

- **SMC':** Combination of the SMC' and the TS would seem to be a fruitful line of future research in the ARG inference problem because the former is the most natural first-order approximation to the CwR (Wilton et al., 2015), and the latter is the current most efficient data structure for the ARG. For moderately large data sets, *ARGinfer* can be extended to use the SMC' instead of the CwR. This modification reduces the computational complexity of the MCMC algorithm by not considering the TNAMs, providing a compromise between accuracy and efficiency. Due to the SMC' structure, it might be possible to divide the inference problem into many sub-tasks suited to parallel architectures. Under the SMC', the *ATS* allows for efficient likelihood and prior evaluation. However, the proposal distribution requires further thought. We must emphasize that it is not trivial if the compactness of the TS would be beneficial in this case.

7.3 Summary

The MCMC algorithm developed in this thesis is the first probabilistic method that uses the TS in the ARG inference problem and samples from the CwR+. *ARGinfer* provides accurate estimates with measures of uncertainty for ARG parameters. Knowledge about these parameters help to learn about many biological processes, e.g., phenotype diversity (Zhang et al., 2002), mapping of disease genes, and genome structure (Arenas, 2013). It is recommended to use *ARGinfer* when a sub-region of the genome is under consideration, and parameters of interest are related to recombination and event times, e.g., the recombination rate, TMRCA, or allele age. It is also useful to extend and use *ARGinfer* in multi-population models to estimate the demographic model, including divergence times and effective population sizes because *ARGinfer* estimates non-ancestral and invisible recombinations that may change populations without changing the tree topologies.

Appendix A

Transition probabilities for the Kuhner move

We prove that (5.9) holds for the Kuhner move for which the MH term reduces to equation (5.10). For simplicity and with no loss of generality, we use the example in section 5.5.6.4. Assume that the ARG presented in Figure 5.9 (a) is G_j and the ARG in Figure 5.10 (d) is G_{j+1}, after applying Kuhner move. The probability of G_j is

$$\begin{aligned} P(G_j) = \frac{\rho}{2} e^{-(1+9\rho)t_1} \times \frac{\rho}{2} e^{-(3+\frac{17\rho}{2})(t_2-t_1)} \times e^{-(6+8\rho)(t_3-t_2)} \\ \times e^{-(3+\frac{21\rho}{2})(t_4-t_3)} \times e^{-(1+7\rho)(t_5-t_4)}. \end{aligned} \tag{A.1}$$

Similarly, the probability of G_{j+1} is given by

$$\begin{aligned} P(G_{j+1}) = \frac{\rho}{2} e^{-(1+9\rho)t_1} \times \frac{\rho}{2} e^{-(3+\frac{17\rho}{2})(t'_2-t_1)} \times e^{-(6+8\rho)(t'_3-t'_2)} \\ \times e^{-(3+\frac{11\rho}{2})(t_4-t'_3)} \times e^{-(1+6\rho)(t'_5-t_4)}. \end{aligned} \tag{A.2}$$

The forward transition probability is

$$\begin{aligned} Q(G_{j+1}|G_j) = & \; e^{-(1+4.5\rho)t_1} && \text{no event in } (0, t_1) \\ & \times \frac{\rho}{2} e^{-(2+4.5\rho)(t'_2-t_1)} && \text{recombination at } t'_2 \in (t_1, t_2) \\ & \times e^{-(1+4+4\rho)(t_2-t'_2)} && \text{no event in } (t'_2, t_2) \\ & \times e^{-(1+4+4\rho)(t_3-t_2)} && \text{no event in } (t_2, t_3) \\ & \times e^{-(1+4+4\rho)(t'_3-t_3)} && \text{CA at } t'_3 \in (t_3, t_4) \\ & \times e^{-(2+3\rho)(t_4-t'_3)} && \text{no event in } (t'_3, t_4) \\ & \times e^{-(1+3\rho)(t_5-t_4)} && \text{no event in } (t_4, t_5) \\ & \times e^{-(1+6\rho)(t'_5-t_5)} && \text{CA at } t'_5 \in (t_5, \infty), \end{aligned} \tag{A.3}$$

and the reverse transition probability is found by

$$\begin{aligned} Q(G_j|G_{j+1}) = & \; e^{-(1+4.5\rho)t_1} && \text{no event in } (0, t_1) \\ & \times e^{-(2+4.5\rho)(t_2'-t_1)} && \text{no event in } (t_1, t_2') \\ & \times \frac{\rho}{2} e^{-(2+4.5\rho)(t_2-t_2')} && \text{recombination at } t_2 \in (t_2', t_3') \\ & \times e^{-(1+4+4\rho)(t_3-t_2)} && \text{CA at } t_3 \in (t_2, t_3') \\ & \times e^{-(2+6.5\rho)(t_3'-t_3)} && \text{no event in } (t_3, t_3') \\ & \times e^{-(2+8\rho)(t_4-t_3')} && \text{no event in } (t_3', t_4) \\ & \times e^{-(1+4\rho)(t_5-t_4)} && \text{CA at } t_5 \in (t_4, t_5'). \end{aligned} \tag{A.4}$$

From (A.1), (A.2), (A.3), and (A.4),

$$\begin{aligned} P(G_{j+1})Q(G_j|G_{j+1}) &= e^{(3+8.5\rho)t_1+(3-0.5\rho)t_2+(-3+3.5\rho)t_3-(3+3.5\rho)t_4} \\ &\quad \times e^{(-1-4\rho)t_5+(5+4\rho)t_2'-(3+\rho)t_3'-(1+6\rho)t_5'} \\ &= P(G_j)Q(G_{j+1}|G_j). \end{aligned} \tag{A.5}$$

Appendix B

List of notation

Table B.1: List of symbols and notations used in this Thesis.

ϕ	the null reference
N	effective population size
μ	mutation rate per site per generation
θ	scaled mutation rate, i.e., $4N\mu$
r	recombination rate per site per generation
ρ	scaled recombination rate, i.e., $4Nr$
Θ	a set of evolutionary parameters, i.e., (μ, r, N)
$L(\Theta)$	the likelihood
n	sample size or the number of DNA sequences
m	number of SNPs
L	sequence Length in sites
D	a set of n DNA sequences comprising L sites, m of which are segregating
D^i	the ith DNA sequence in D and $i = 1, 2, ..., n$
D^i_j	the allele at the jth SNP of D^i, $j = 1, 2, ..., m$
G	all possible ARGs for D
k	number of lineages in a time interval in an ARG
k'	number of recombination links in a time interval in an ARG.
G_j	a realization of the CwR. Occasionally, the ARG at the jth MCMC iteration
B_j	number of branches on G_j, excluding the roots
N_r	number of recombination events in an ARG
N_c	number of CA events in an ARG
$X(j)$	the state of the MCMC after j iterations
$Q(.)$	a proposal distribution
$Q*(n, \rho)$	probability that n sequences that share a TMRCA at point 0 and 1 do not share one in at least one intervening interval, given ρ
b	recombination breakpoint

Appendix C

List of abbreviations

Table C.1: List of abbreviations used in this Thesis.

ARG	ancestral recombination graph
ATS	augmented tree sequences
CA	common ancestor event
CC	compatibility check
CR	coalescence record
CwR	coalescent with recombination
DSMC	discretized sequentially Markov coalescent
ESS	effective sample size
FTS	full tree sequences
GMRCA	great most recent common ancestor
HWE	Hardy-Weinberg Equilibrium
ISM	infinite sites model
JC69	Jukes-Cantor mutation model
KC	Kendall-Colijn metric
LD	linkage disequilibrium
MAF	minor allele frequency
MCMC	Markov chain Monte Carlo
$(MC)^3$	Metropolis coupled MCMC
MH	Metropolis-Hastings
MRCA	most recent common ancestor
NAM	no ancestral material
RMSE	root mean square error
SMC	sequentially Markov coalescent
SNP	single nucleotide polymorphism
SPR	subtree-prunning-and-regrafting
TMRCA	time to the most recent common ancestor
TNAM	trapped non-ancestral ancestral material
TS	tree sequences
VC	validity check

Bibliography

1000 Genomes Project Consortium (2015). A global reference for human genetic variation. *Nature*, 526(7571):68–74.

Albers, P. K. and McVean, G. (2020). Dating genomic variants and shared ancestry in population-scale sequencing data. *PLoS Biology*, 18(1):e3000586.

Arenas, M. (2013). The importance and application of the ancestral recombination graph. *Frontiers in Genetics*, 4:206.

Bycroft, C., Freeman, C., Petkova, D., Band, G., Elliott, L. T., Sharp, K., Motyer, A., Vukcevic, D., Delaneau, O., O'Connell, J., et al. (2018). The uk biobank resource with deep phenotyping and genomic data. *Nature*, 562(7726):203–209.

Chen, G. K., Marjoram, P., and Wall, J. D. (2009). Fast and flexible simulation of DNA sequence data. *Genome Research*, 19(1):136–142.

Danecek, P., Auton, A., Abecasis, G., Albers, C. A., Banks, E., DePristo, M. A., Handsaker, R. E., Lunter, G., Marth, G. T., Sherry, S. T., et al. (2011). The variant call format and vcftools. *Bioinformatics*, 27(15):2156–2158.

Dinh, V., Bilge, A., Zhang, C., and Matsen IV, F. A. (2017). Probabilistic path hamiltonian monte carlo. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1009–1018.

Donnelly, P. and Leslie, S. (2010). The coalescent and its descendants. *arXiv preprint arXiv:1006.1514*.

Ewing, G. and Hermisson, J. (2010). Msms: a coalescent simulation program including recombination, demographic structure and selection at a single locus. *Bioinformatics*, 26(16):2064–2065.

Fearnhead, P. and Donnelly, P. (2001). Estimating recombination rates from population genetic data. *Genetics*, 159(3):1299–1318.

Felsenstein, J., Kuhner, M. K., Yamato, J., and Beerli, P. (1999). Likelihoods on coalescents: a monte carlo sampling approach to inferring parameters from population samples of molecular data. *Lecture Notes-Monograph Series*, pages 163–185.

Forney, G. D. (1973). The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Griffiths, R. C. and Marjoram, P. (1996). Ancestral inference from samples of dna sequences with recombination. *Journal of Computational Biology*, 3(4):479–502.

Griffiths, R. C. and Marjoram, P. (1997). An ancestral recombination graph. *Institute for Mathematics and its Applications*, 87:257.

Griffiths, R. C. and Tavaré, S. (1994). Simulating probability distributions in the coalescent. *Theoretical Population Biology*, 46(2):131–159.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications.

Hein, J. (1993). A heuristic method to reconstruct the history of sequences subject to recombination. *Journal of Molecular Evolution*, 36(4):396–405.

Hein, J., Schierup, M., and Wiuf, C. (2004). *Gene genealogies, variation and evolution: a primer in coalescent theory*. Oxford University Press, USA.

Heine, K., Beskos, A., Jasra, A., Balding, D., and De Iorio, M. (2018). Bridging trees for posterior inference on ancestral recombination graphs. *Proceedings of the Royal Society A*, 474(2220):20180568.

Hellenthal, G. and Stephens, M. (2007). msHOT: modifying Hudson's ms simulator to incorporate crossover and gene conversion hotspots. *Bioinformatics*, 23(4).

Hubisz, M. (2019). Inferring the population history of ancient hominins through use of the ancestral recombination graph.

Hudson, R. R. (1983). Properties of a neutral allele model with intragenic recombination. *Theoretical Population Biology*, 23(2):183–201.

Hudson, R. R. (2002). Generating samples under a wright–fisher neutral model of genetic variation. *Bioinformatics*, 18(2):337–338.

Hudson, R. R. and Kaplan, N. L. (1985). Statistical properties of the number of recombination events in the history of a sample of DNA sequences. *Genetics*, 111(1):147–164.

Jenkins, P. A. (2008). Importance sampling on the coalescent with recombination.

Jukes, T. H., Cantor, C. R., Munro, H., et al. (1969). Evolution of protein molecules. *Mammalian Protein Metabolism*, 3(21):132.

Kass, R. E., Carlin, B. P., Gelman, A., and Neal, R. M. (1998). Markov chain monte carlo in practice: a roundtable discussion. *The American Statistician*, 52(2):93–100.

Kececioglu, J. and Gusfield, D. (1998). Reconstructing a history of recombinations from a set of sequences. *Discrete Applied Mathematics*, 88(1-3):239–260.

Kelleher, J., Etheridge, A. M., and McVean, G. (2016). Efficient coalescent simulation and genealogical analysis for large sample sizes. *PLoS Computational Biology*, 12(5):e1004842.

Kelleher, J., Thornton, K. R., Ashander, J., and Ralph, P. L. (2018). Efficient pedigree recording for fast population genetics simulation. *PLoS Computational Biology*, 14(11):e1006581.

Kelleher, J., Wong, Y., Wohns, A. W., Fadil, C., Albers, P. K., and McVean, G. (2019). Inferring whole-genome histories in large population datasets. *Nature Genetics*, 51(9):1330–1338.

Kendall, M. and Colijn, C. (2016). Mapping phylogenetic trees to reveal distinct patterns of evolution. *Molecular Biology and Evolution*, 33(10):2735–2743.

Kimura, M. (1969). The number of heterozygous nucleotide sites maintained in a finite population due to steady flux of mutations. *Genetics*, 61(4):893.

Kingman, J. F. (1982a). On the genealogy of large populations. *Journal of Applied Probability*, 19(A):27–43.

Kingman, J. F. C. (1982b). The coalescent. *Stochastic Processes and Their Applications*, 13(3):235–248.

Koskela, J. (2020). Zig-zag sampling for discrete structures and non-reversible phylogenetic mcmc. *arXiv preprint arXiv:2004.08807*.

Kreitman, M. (1983). Nucleotide polymorphism at the alcohol dehydrogenase locus of drosophila melanogaster. *Nature*, 304(5925):412–417.

Kuhner, M. K., Yamato, J., and Felsenstein, J. (1995). Estimating effective population size and mutation rate from sequence data using metropolis-hastings sampling. *Genetics*, 140(4):1421–1430.

Kuhner, M. K., Yamato, J., and Felsenstein, J. (2000). Maximum likelihood estimation of recombination rates from population data. *Genetics*, 156(3):1393–1401.

Lafayette, L., Sauter, G., Vu, L., and Meade, B. (2016). Spartan performance and flexibility: An hpc-cloud chimera. *OpenStack Summit, Barcelona*, 27.

Li, N. and Stephens, M. (2003). Modeling linkage disequilibrium and identifying recombination hotspots using single-nucleotide polymorphism data. *Genetics*, 165(4):2213–2233.

Marjoram, P., Markovtsova, L., and Tavaré, S. (2000). Estimation in ancestral recombination graphs using markov chain monte carlo. Technical report, Research Report.

Marjoram, P. and Wall, J. D. (2006). Fast "coalescent" simulation. *BMC Genetics*, 7(1):16.

McVean, G. A. and Cardin, N. J. (2005). Approximating the coalescent with recombination. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 360(1459):1387–1393.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092.

Minichiello, M. J. and Durbin, R. (2006). Mapping trait loci by use of inferred ancestral recombination graphs. *The American Journal of Human Genetics*, 79(5):910–922.

Mirzaei, S. and Wu, Y. (2017). Rent+: an improved method for inferring local genealogical trees from haplotypes with recombination. *Bioinformatics*, 33(7):1021–1030.

Nguyen, T. T. P., Le, V. S., Ho, H. B., and Le, Q. S. (2017). Building ancestral recombination graphs for whole genomes. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 14(2):478–483.

Nielsen, R. (2000). Estimation of population parameters and recombination rates from single nucleotide polymorphisms. *Genetics*, 154(2):931–942.

Nordborg, M. (2019). Coalescent theory. *Handbook of Statistical Genomics: Two Volume Set*, pages 145–30.

Purcell, S., Neale, B., Todd-Brown, K., Thomas, L., Ferreira, M. A., Bender, D., Maller, J., Sklar, P., De Bakker, P. I., Daly, M. J., et al. (2007). Plink: a tool set for whole-genome association and population-based linkage analyses. *The American Journal of Human Genetics*, 81(3):559–575.

Ralph, P., Thornton, K., and Kelleher, J. (2020). Efficiently summarizing relationships in large samples: a general duality between statistics of genealogies and genomes. *Genetics*.

Rasmussen, M. D., Hubisz, M. J., Gronau, I., and Siepel, A. (2014). Genome-wide inference of ancestral recombination graphs. *PLoS Genetics*, 10(5):e1004342.

Song, Y. S., Wu, Y., and Gusfield, D. (2005). Efficient computation of close lower and upper bounds on the minimum number of recombinations in biological sequence evolution. *Bioinformatics*, 21(suppl_1):i413–i422.

Speidel, L., Forest, M., Shi, S., and Myers, S. R. (2019). A method for genome-wide genealogy estimation for thousands of samples. *Nature Genetics*, 51(9):1321–1329.

Staab, P. R., Zhu, S., Metzler, D., and Lunter, G. (2015). Scrm: efficiently simulating long sequences using the approximated coalescent with recombination. *Bioinformatics*, 31(10):1680–1682.

Stephens, M. and Donnelly, P. (2000). Inference in molecular population genetics. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(4):605–635.

Tajima, F. (1983). Evolutionary relationship of dna sequences in finite populations. *Genetics*, 105(2):437–460.

Wakeley, J. (1997). Using the variance of pairwise differences to estimate the recombination rate. *Genetics Research*, 69(1):45–48.

Wakeley, J. (2009). *Coalescent theory: an introduction*. Number 575: 519.2 WAK.

Wakeley, J. (2020). Developments in coalescent theory from single loci to chromosomes. *Theoretical Population Biology*.

Wall, J. D. (2000). A comparison of estimators of the population recombination rate. *Molecular Biology and Evolution*, 17(1):156–163.

Wang, Y., Zhou, Y., Li, L., Chen, X., Liu, Y., Ma, Z.-M., and Xu, S. (2014). A new method for modeling coalescent processes with recombination. *BMC Bioinformatics*, 15(1):273.

Watterson, G. (1975). On the number of segregating sites in genetical models without recombination. *Theoretical Population Biology*, 7(2):256–276.

Wilson, I. J. and Balding, D. J. (1998). Genealogical inference from microsatellite data. *Genetics*, 150(1):499–510.

Wilton, P. R., Carmi, S., and Hobolth, A. (2015). The smc' is a highly accurate approximation to the ancestral recombination graph. *Genetics*, 200(1):343–355.

Wiuf, C. and Hein, J. (1999). Recombination as a point process along sequences. *Theoretical Population Biology*, 55(3):248–259.

Wiuf, C. and Hein, J. (2000). The coalescent with gene conversion. *Genetics*, 155(1):451–462.

Wu, Y. (2009). New methods for inference of local tree topologies with recombinant snp sequences in populations. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(1):182–193.

Zhang, Y.-X., Perry, K., Vinci, V. A., Powell, K., Stemmer, W. P., and del Cardayré, S. B. (2002). Genome shuffling leads to rapid phenotypic improvement in bacteria. *Nature*, 415(6872):644–646.